University of Alberta

DEVELOPMENT AND APPLICATIONS OF TOOLS FOR COMPUTATIONAL STUDIES OF VERY LARGE MOLECULAR SYSTEMS

by

Jonathan Y. Mane ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**

Department of Chemistry

Edmonton, Alberta
Fall 2008

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-46377-2
Our file Notre référence
ISBN: 978-0-494-46377-2

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Dedicated to my loving and caring wife Maricel

Abstract

Modern computational chemistry utilizes the principles of theoretical chemistry to calculate molecular properties, and predict physical and chemical behavior of molecular systems. Methods that are used in computational chemistry can range from the relatively computationally inexpensive molecular mechanics (MM) to highly advanced but computationally expensive quantum mechanical (QM) techniques. Molecular mechanics uses fixed charges in calculations even if the molecular system changes configuration. Quantum mechanics, on the other hand, calculates charges dynamically in response to changes in the configuration of the system. However, as the molecular system gets larger, calculation using QM techniques becomes prohibitively expensive. To take advantage of the relative speed of MM calculations and the accuracy that is provided by using dynamic charges from QM calculations, a hybrid QM/MM technique is often employed.

The development of computational tools for modeling large complex systems in the context of using a hybrid QM/MM methodology is the inspiration in this work. Two approaches have been taken in order to accomplish such tools development. The first approach involved the transferability of molecular properties obtained from a high level QM calculation to a lower level QM method like the semi-empirical method. This was accomplished through the optimization of semi-empirical parameters, via the genetic algorithm (GA),

using molecular properties obtained from MP2 calculations as the reference. The GA optimized parameters were then used as the new parameters for the semi-empirical method to test how well they represent the molecular properties of the MP2 reference. The second approach involved an on-the-fly QM calculation of molecular properties. This was done by extending the capability of the DYNAMO program to use advanced quantum mechanical potentials found in GAMESS-US, in addition to its built-in semi-empirical potentials.

The DYNAMO program, which is a library of FORTRAN 90 modules capable of performing hybrid QM/MM calculations, was extensively used in this work both for tools development and applications. The DYNAMO library has been used in the calculation of free energies of binding for colchicine and its derivatives with different α/β-tubulin isoforms. Many examples were also provided on how to use the DYNAMO library for performing molecular simulations.

Acknowledgements

I would like to thank the following persons during my stay at the Department of Chemistry, University of Alberta: Dr. Mariusz Klobukowski, my research supervisor, for his excellent guidance and support, and for providing me the opportunity to be part of his research group to be trained as a computational chemist. Dr. Anna Jordan, whom I worked for while teaching Physical Chemistry Laboratory. Christopher Lovallo, John Lo, Melissa Gajewski, Toby Zeng, Evan Kelly and Jesse Kadosh, my friends in MK Group. I would also like to thank Dr. Jack Tuszynski and Dr. J. Torin Huzil of the Department of Oncology, University of Alberta.

I would also like to acknowledge the University of Alberta Department of Chemistry for my TA scholarship, the Natural Sciences and Engineering Research Council (NSERC) of Canada for the funding of workstations, the Mathematics of Information Technology and Complex Systems (MITACS) for funding and support, the University of Alberta Academic Information and Communication Technology (AICT) for the IBM RS/6000 workstations and for the General Purpose Linux cluster, WestGrid for computers, and The Allard Foundation, Oncovista, LLC of San Antonio, TX and Technology Innovations, LLC of Rochester, NY for additional support.

Table of Contents

1 Introduction **1**
1.1 *Ab initio* Molecular Orbital Methods 3
1.1.1 Hartree-Fock Method 3
1.1.2 Post Hartree-Fock Methods 10
1.1.3 Density Functional Theory 12
1.2 Basis Sets 16
1.2.1 Slater and Gaussian Type Functions 16
1.3 The Semi-empirical SCF MO and the NDDO Method 18
1.3.1 MNDO Method 22
1.3.2 The AM1 and PM3 Methods 23
1.3.3 The Semi-empirical Parameters 24
1.4 Classical Molecular Mechanics Method 25
1.5 Hybrid Quantum Mechanics and Molecular Mechanics Methods 26
1.5.1 The Link Atom Approach 28
1.6 Derivatives of the Energy Function 30
1.7 Energy Minimization and Simulated Annealing 31
1.8 Molecular Dynamics 32
1.8.1 The Integration Algorithms 33
1.8.2 Performing Molecular Dynamics Simulations 35
1.8.3 NVE, NVT and NPT Molecular Dynamics 37
1.9 Hybrid QM/MM–Molecular Dynamics 39
1.10 Solvents and the Free Energy of Solvation 40

2 Genetic Algorithm: Theory and Programming **42**
2.1 The Genetic Algorithm 44
2.1.1 Chromosome Representation 44
2.1.2 Creation of the Initial Population 45
2.1.3 The Fitness Function 45
2.1.4 Selection Operation 46
2.1.5 Reproduction or Crossover Operation 47
2.1.6 Mutation Operation 49
2.1.7 Termination of GA 49

2.2 Program Description and Coding 50
  2.2.1 The Zeroth Generation 50
  2.2.2 The Subsequent Generations 54
  2.2.3 The GA Termination 56
2.3 Conclusion 59

3 Genetic Algorithm: Application to Semi-empirical PM3 parameters for Monosaccharides 61
3.1 Computational Methods 63
3.2 Results and Discussion 65
3.3 Conclusion 70

4 Free Energy Calculations: Theory 72
4.1 Methods to Calculate Free Energies 73
  4.1.1 The Free Energy Perturbation 74
  4.1.2 The Umbrella Sampling 78
4.2 Conclusion 82

5 Cancer Cells, Chemotherapy Drugs and Microtubules 83
5.1 The Cell 86
5.2 The Cytoskeleton and the Filamentous Protein Assemblies . . 88
5.3 Microtubules and the Cell Cycle 89
5.4 Microtubules, Tubulin, and Colchicine 92
  5.4.1 Microtubules 92
  5.4.2 Tubulin isoforms and their 3D Models 92
  5.4.3 Colchicine and Other Anti-mitotic Agents 98
  5.4.4 The Colchicine Binding Site 101
5.5 Conclusion 104

6 Free Energy Calculations: Application to the Relative Binding Free Energy of Colchicine and Its Derivatives with the α/β-tubulin Isoforms 105
6.1 Relative binding energy calculations 106
  6.1.1 Setting up the initial structures 106
  6.1.2 Free energy of binding calculations and their results . . 109
6.2 Conclusion 117

7 DYNAMO and Umbrella Sampling: An Application to Diaminopimelate (DAP) 119
7.1 Umbrella sampling to calculate the free energy profile 120
  7.1.1 Setting up the system 120
  7.1.2 The reaction coordinate for the free energy profile . . . 122

7.1.3 A program illustrating the calculation of the free energy profile . . . 122
7.2 Conclusion . . . 123

8 The DYNAMO Library and the GAMESS-US Program 129
8.1 FORTRAN 90 . . . 130
8.1.1 The Main Program Structure . . . 130
8.1.2 Module Structure . . . 130
8.2 The DYNAMO Module Library . . . 132
8.2.1 The Dynamo Coordinate, Sequence, and Force Field Files 133
8.3 The GAMESS-US Program Input File . . . 133
8.4 The GAMESS-US Modules for DYNAMO . . . 137
8.4.1 The GAMESS-US Data Module . . . 138
8.4.2 The GAMESS-US Controls Module . . . 140
8.4.3 The GAMESS-US Coordinates . . . 140
8.4.4 The GAMESS-US Hamiltonian Module . . . 141
8.4.5 The GAMESS-US Energy Module . . . 142
8.4.6 The GAMESS-US Gradients Module . . . 143
8.5 Modifications to DYNAMO Modules . . . 143
8.6 An Example of Using DYNAMO/GAMESS-US Library . . . 144
8.7 Conclusion . . . 146

9 Summary and Conclusions 147

Bibliography 150

A Miscellaneous Topics: Examples for Using DYNAMO Library 168
A.1 Addition of hydrogen coordinates: An example using α/β-tubulin coordinates . . . 168
A.2 Solvating the system: An example of solvating colchicine-α/β-tubulin system . . . 169
A.3 A molecular dynamics simulation: Langevin dynamics . . . 171
A.4 A protocol for simulated annealing: An application to metal–rare gas clusters . . . 174

List of Tables

1.1 Scaling behavior of some *ab initio* methods. 19
1.2 Semi-empirical parameters for MNDO, AM1, and PM3. 24

3.1 $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for hydrogen. . . 66
3.2 $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for carbon. . . . 66
3.3 $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for oxygen. . . . 67
3.4 Relative energies (in kJ/mol) of the different ${}^4C_1$ conformations of D-galactose, D-glucose and D-mannose and the r.m.s. deviations with respect to MP2/cc-pVTZ. 68
3.5 S_{rms} deviations (in Å) for the ${}^4C_1$ conformations of D-galactose, D-glucose and D-mannose with respect to MP2/cc-pVTZ geometry. 69
3.6 Relative energies (in kJ/mol) of the different ${}^4C_1$ conformations of pentapyranoses and the average r.m.s. deviations with respect to MP2/cc-pVTZ. 70
3.7 S_{rms} deviations (in Å) of the different ${}^4C_1$ conformations of pentapyranoses with respect to MP2/cc-pVTZ. 71

5.1 Some commercially available chemotherapy drugs. 85
5.2 Comparison between prokaryotic and eukaryotic cells. 87
5.3 The organelles in eukaryotic cells and their function. 88
5.4 Tissue distribution of β-tubulin isoforms in normal cells. . . . 96
5.5 Tubulin isoform interactions with colchicine. 102

6.1 Calculated average $\Delta\Delta G_{bind}$ and standard deviations (in kcal/mol) for colchicine derivatives **1-16** against colchicine in 10 α/β-tubulin dimers. 113
6.2 Calculated average $\Delta\Delta G_{bind}$ and standard deviations (in kcal/mol) for colchicine derivatives **17-22** against colchicine in 10 α/β-tubulin dimers. 114

8.1 The GAMESS-US modules added to the DYNAMO module library. 138
8.2 The modified modules in the original DYNAMO distribution. . 144

List of Figures

1.1 A general algorithm for molecular dynamics simulation. 36

2.1 An outline of algorithm for a GA program. 44
2.2 A binary encoding of the chromosome. 45
2.3 A real-valued encoding of the chromosome. 45
2.4 A single-point crossover. 48
2.5 A two-point crossover. 48
2.6 A uniform crossover. 49
2.7 A mutation in the binary encoding of the chromosome. 50
2.8 A mutation in the real-valued encoding of the chromosome. . . 50
2.9 The pseudocode for the main GA program. 51
2.10 An example of the structure of the starter file. 52
2.11 An example of a generated initial population. 52
2.12 The labels corresponding to each of the PM3 parameters used in GA. All parameters are placed in a single line. 53
2.13 The code for the crossover of the GA program. 55
2.14 The code for the mutation of the elite individuals of the GA program. 57
2.15 The code for the mutation of the other individuals of the GA program. 58
2.16 The code for calculating the standard deviation in the GA program. 60

3.1 The structures of the D-hexopyranoses. 63
3.2 The structures of the D-pentapyranoses. 65

4.1 The thermodynamic cycle. 78

5.1 The cell cycle. 90
5.2 Typical structures of microtubule. 93
5.3 Model representation of α/β-tubulin heterodimer. 94
5.4 Vinblastine binding site located at the inter-tubulin dimer interface. 99
5.5 α/β-tubulin heterodimers showing the taxane binding site. . . 100

5.6 Schematic diagram of isoform differences in the colchicine binding site. Models of the βIII, βV, βVI, βVII and βVIII isoforms were superimposed, along with the β chain B from 1SA0[1], and residues containing any atom within 6 Å of the bound colchicine molecule were selected. The amino side chain differences between each of the isoforms within this cut-off are shown as red sticks, while colchicine is shown in green sticks. (A) shows those residues, as they would be observed within the β tubulin consensus sequence. (B) is an identical view as (A), with the exception that each position carries the appropriate amino acid substitution as seen in the isoforms. 103

6.1 The structure of colchicine. 107
6.2 The structure of the colchicine analogue and the corresponding X-side groups. 108
6.3 The structure of the colchicine analogue and the corresponding Y-side groups. 109
6.4 The thermodynamic cycle. 110
6.5 α/β-tubulin isoforms and compounds **1-16**. 115
6.6 α/β-tubulin isoforms and compounds **17-22**. 116

7.1 The epimerization reaction of L,L-DAP. 120
7.2 The structure of L,L-DAP used in the umbrella sampling in the presence of Cys73–S$^-$ and Cys217–S–H pair. 121
7.3 Program illustrating the generation of windows for calculating the free energy profile. 124
7.4 Program illustrating the generation of windows for calculating the free energy profile (contd). 125
7.5 A program to perform WHAM. 126
7.6 The energy profile for L,L-DAP. 127

8.1 FORTRAN 90 program structure. 131
8.2 FORTRAN 90 module structure. 131
8.3 An example of user-written FORTRAN 90 program. 134
8.4 DYNAMO "`water.opls`" force field file. 135
8.5 DYNAMO "`water_dimer.seq`" sequence file. 136
8.6 DYNAMO "`water_dimer.crd`" coordinate file. 136
8.7 An example of GAMESS-US input file. 137

A.1 Program for building the hydrogen coordinates. 169
A.2 A portion of the `ab_tub_I.seq` file. 170
A.3 Program for solvating the colchicine-α/β-tubulin complex in a rectangular box of water molecules. 172
A.4 Program for performing Langevin molecular dynamics simulation.173

A.5 Program for performing a simulated annealing on a cluster of one strontium and 24 helium atoms. 175
A.6 Program for performing a simulated annealing on a cluster of one strontium and 24 helium atoms (contd). 176
A.7 The subroutine `WRITE_STRUCTURE`. 177
A.8 The `M_He.opls` file. 178
A.9 The `sr_he_24.seq` file. 179

List of Abbreviations

AE	All-electron
AM1	Austin Model 1
AO	Atomic orbital
B3LYP	Becke's three parameter hybrid functional with the Lee-Yang-Parr correlation functional for use with DFT
cc-pVDZ	Correlation-consistent polarized valence double zeta
cc-pVTZ	Correlation-consistent polarized valence triple zeta
cGTF	Contracted Gaussian-type function
CI	Configuration interaction
CNDO	Complete neglect of differential overlap
COSMO	Conductor-like screening model
DNA	Deoxyribonucleic acid
DFT	Density functional theory
ES	Evolution strategy
EP	Evolutionary programming
FEP	Free energy perturbation
GA	Genetic algorithm
GTF	Gaussian-type function
HF	Hartree-Fock
HFRH	Hartree-Fock-Roothaan-Hall
IF	Intermediate filament
IV	Intravenous
INDO	Intermediate neglect of differential overlap
KS	Kohn-Sham
LDA	Local density approximation
MC	Monte Carlo
MD	Molecular dynamics
MM	Molecular mechanics
MNDO	Modified neglect of diatomic overlap
MO	Molecular orbital
MP2	Second-order Møller-Plesset perturbation
MP3	Third-order Møller-Plesset perturbation

MT	Microtubule
MTOC	Microtubule organizing center
NDDO	Neglect of diatomic differential overlap
PCM	Polarizable continuum model
*p*GTF	Primitive Gaussian-type function
PM3	Parameterization 3
PM6	Parameterization 6
PMF	Potential of mean force
QM	Quantum mechanics
QMMM	Hybrid quantum mechanics and molecular mechanics
QMMD	Hybrid quantum mechanics and molecular dynamics
RHF	Restricted Hartree-Fock
RM1	Recife Model 1
RNA	Ribonucleic acid
ROHF	Restricted open-shell Hartree-Fock
SCF	Self-consistent field
SCRF	Self-consistent reaction field
STF	Slater-type function
STO	Slater type orbital
STO–nG	Minimal basis set with $n-$Gaussian functions used to represent the atomic orbitals on an atom
XC	Exchange-correlation
ZDO	Zero differential overlap

Chapter 1

Introduction

Computational chemistry has become one of the most popular tools in modern chemistry. It incorporates not only the theoretical models and principles of quantum mechanics that have been developed over the past eight decades, but also computer-based methods for understanding, interpreting and/or predicting the behavior of molecular systems that were developed more recently. Computer programs and algorithms, molecular modeling, and methods such as minimization or optimization, simulations, molecular mechanics, molecular dynamics and conformational analysis are just few of the terms often associated with computational chemistry. With today's improvements in computer hardware capabilities and a high performance-to-cost ratio of computers, simulations of experiments using computers are now very practical and have advantages – reduction in operating expenses and increased safety in doing experiments by avoiding expensive and toxic chemicals. By using computers, it is possible to simulate experiments which may be very difficult or even impossible to do in even the most sophisticated laboratory. Although some simulated computer experiments may not be achievable in the laboratory, they may be useful for testing methods by finding out the importance of various effects (e.g. electron correlation, relativistic effects, etc.).

Computational chemistry has become a widely used tool and an integral part of scientific investigation. It often works side by side with experimental chemistry, such as X-ray crystallography and NMR spectroscopy, to develop better theoretical models with more accurate predictive power. The quality of results achieved from computational or theoretical calculations relies heavily on the methods employed. In general, theoretical calculations are often divided into two major classes—*empirical* or *non-empirical*. Empirical calculations employ classical molecular mechanical (MM) methodologies and techniques such as classical molecular dynamics (MD). Non-empirical methods make use of quantum mechanical (QM) methods such as *ab initio* molecular orbital (MO) theory, semi-empirical MO approaches, and density functional theory. Depending on the size and complexity of the molecular model system being studied, a combination of both empirical and non-empirical methods may also be used. This is often referred to as hybrid QM/MM.

The prediction and determination of accurate geometries and electron distribution properties of molecular systems not only depends on the computational methods used, but also on certain numerical values or parameters used in the actual computation. These parameters are referred to as basis sets or basis functions in quantum mechanical calculations and force-fields in classical molecular mechanics. Achieving quality results requires the use of high quality parameters.

The problem of obtaining high quality parameters is addressed in this work. This involves the development computational tools for modeling large complex systems in the context of hybrid QM/MM methodology. Two approaches have been taken in order to accomplish the task. The first approach involved the transferability of molecular properties obtained from a high level QM calculation to a lower level QM method like the semi-empirical method by means of optimization techniques and fitting procedures. The second approach involved an on-the-fly high level QM calculation of molecular properties in conjunction with MM methodologies. The details are laid out in later chapters of this thesis.

In this chapter, an introduction to the theories and methods relevant to this work is discussed. These include a simple description of *ab initio* methods, particularly Hartree-Fock theory and post Hartee-Fock methods, followed by a discussion on semi-empirical methods. A brief description of classical molecular mechanics and molecular dynamics is then presented followed by a brief discussion on hybrid quantum mechanical and molecular mechanical approaches and solvent effects.

1.1 *Ab initio* Molecular Orbital Methods

1.1.1 Hartree-Fock Method

The detailed description and derivation of formulas presented in this section can be found in many standard quantum chemical textbooks. The major references used here include the books by Szabo and Ostlund [2], Levine [3], Jensen [4] and McQuarrie [5]. One of the major challenges facing quantum chemists is finding the exact solution to the non-relativistic time-independent Schrödinger equation

$$\hat{H}\Psi(R_A, r_i) = E\Psi(R_A, r_i) \tag{1.1}$$

which yields, upon solving, the wavefunctions Ψ and energies E. In Eq. (1.1), $\hat{H}$ is the Hamiltonian operator for a system of nuclei and electrons described by the position vectors R_A and r_i, respectively, and is given in atomic units as

$$\begin{aligned} \hat{H} = & \underbrace{-\sum_{A=1}^{M} \frac{1}{2M_A}\nabla_A^2}_{\hat{T}_N} \underbrace{-\sum_{i=1}^{N}\frac{1}{2}\nabla_i^2}_{\hat{T}_e} \\ & \underbrace{-\sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}}}_{\hat{V}_{Ne}} \underbrace{+\sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}}}_{\hat{V}_{ee}} \underbrace{+\sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}}}_{\hat{V}_{NN}} . \end{aligned} \tag{1.2}$$

The first term in Eq. (1.2) is the operator for the kinetic energy of the nuclei, $\hat{T}_N$. The second term is the operator for the kinetic energy of the electrons, $\hat{T}_e$. The third term represents the Coulomb attraction between the electrons and the nuclei, $\hat{V}_{Ne}$, where r_{iA} is the distance between electron i and nucleus A. The fourth and fifth terms are $\hat{V}_{ee}$, the potential energy of repulsions between electrons i and j separated by a distance r_{ij}, and $\hat{V}_{NN}$, the potential energy of repulsions between nuclei A and B separated by a distance $R_{AB} = |\vec{R}_A - \vec{R}_B|$, respectively.

Eq. (1.1) is very difficult to solve and approximations must be made. The key to simplifying the solution to Eq. (1.1) is to separate the electronic from the nuclear degrees of freedom. This approximation, known as the *Born-Oppenheimer approximation*, says that the true molecular wavefunction can be approximated as:

$$\Psi(R_A, r_i) = \Psi_{el}(r_i; R_A)\Psi_N(R_A). \tag{1.3}$$

The electrons are much lighter and so they move much faster than the nuclei. During the entire electronic motion, the nuclei move very slowly and the effects of nuclear motion can be considered negligible. Hence, we can think of the electrons in a molecule as moving in the field of fixed nuclei. Within the Born-Oppenheimer approximation, $\hat{T}_N$ can be neglected and $\hat{V}_{NN}$ can be considered constant since the nuclear positions are fixed. Since $\hat{V}_{NN}$ is a constant, it does not affect the electronic wavefunction. Therefore, we can write the electronic Schrödinger equation as

$$(\hat{H}_{el} + \hat{V}_{NN})\Psi_{el} = E_{tot}\Psi_{el} \tag{1.4}$$

where

$$\begin{aligned}
\hat{H}_{el} &= -\sum_{i=1}^{N}\frac{1}{2}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}} && (1.5)\\
\hat{V}_{NN} &= \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{R_{AB}} && (1.6)\\
E_{tot} &= E_{el} + V_{NN}. && (1.7)
\end{aligned}$$

Omitting $\hat{V}_{NN}$ from Eq. (1.4), a purely electronic Schrödinger equation is obtained, *i.e.*

$$\hat{H}_{el}\Psi_{el} = E_{el}\Psi_{el}. \tag{1.8}$$

Ψ_{el} is the electronic wavefunction describing the motion of N electrons in the field of M point charges. Both Ψ_{el} and E_{el} depend *parametrically* on the nuclear configuration:

$$\begin{aligned}
\Psi_{el} &= \Psi_{el}(\{R_A\}) && (1.9)\\
E_{el} &= E_{el}(\{R_A\}). && (1.10)
\end{aligned}$$

This means that for different arrangements of the nuclei, there are different values for Ψ_{el} and E_{el}. Equations (1.9) and (1.10) make it possible to define the potential energy surface of a molecule as a function of nuclear coordinates.

Although Eq. (1.5) is simplified, it still contains a many-body Hamiltonian and is very difficult to solve. Analytic solutions exist only for systems having one electron. The *Hartree-Fock (HF) method* offers a very popular approach to solving the many-body eigenvalue problem. The method allows to transform the full N-body equation into N single-body equations.

For a closed-shell system involving electrons, the ground-state Hartree-Fock wavefunction is given by a single antisymmeterized function, the Slater determinant, of N spinorbitals, $\psi_i(\mathbf{x}_j)$:

$$\Psi_0 = \frac{1}{\sqrt{N!}} \begin{vmatrix} \psi_1(\mathbf{x}_1) & \psi_1(\mathbf{x}_2) & \dots & \psi_1(\mathbf{x}_N) \\ \psi_2(\mathbf{x}_1) & \psi_2(\mathbf{x}_2) & \dots & \psi_2(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_N(\mathbf{x}_1) & \psi_N(\mathbf{x}_2) & \dots & \psi_N(\mathbf{x}_N) \end{vmatrix} \tag{1.11}$$

where the variable $\mathbf{x}$ corresponds to space, $\mathbf{r}$, and spin coordinates, σ. In this notation, each column of the determinant refers exclusively to the jth electron described by different spinorbitals, and each row refers solely to the ith spinorbital describing distinct electrons. The spinorbital, $\psi_i(\mathbf{x}_j)$, is a simple product of space and spin functions, α and β:

$$\psi_i(\mathbf{x}_j) = \phi_i(\mathbf{r}_j) \times \begin{cases} \alpha(\sigma_j) \\ \beta(\sigma_j) \end{cases} . \tag{1.12}$$

According to the variational principle, the best wavefunction Ψ_0 is the one that minimizes the ground state energy, E_0:

$$E_0 = \left\langle \Psi_0 \middle| \hat{H}_{el} \middle| \Psi_0 \right\rangle . \tag{1.13}$$

Using the wavefunction in Eq. (1.11) in Eq. (1.13), the energy E_0 becomes a functional of the spinorbitals, $\psi_i(\mathbf{x}_j)$. Finding the best spinorbitals which minimize E_0 corresponds to finding the best possible approximation to the ground state of the N-electron system described by $\hat{H}_{el}$. Such an approximation, which reduces the problem of N-electron equations to a set of 1-electron eigenvalue equations, is known as the Hartree-Fock equation:

$$\hat{F}(\mathbf{x}_1)\psi_i(\mathbf{x}_1) = \epsilon_i\psi_i(\mathbf{x}_1), \qquad i = 1, 2, \ldots, N. \tag{1.14}$$

Furthermore, by exploiting the orthonormality of the spinorbitals, the HF equation can be written in terms of spatial orbitals:

$$\hat{F}(1)\phi_i(1) = \epsilon_i\phi_i(1), \qquad i = 1, 2, \ldots, N \tag{1.15}$$

where the label '1' is used to represent the spatial coordinates of a single electron under consideration, even though the actual electron may not be 'electron 1'. In Eq. (1.15), $\hat{F}(1)$ is the effective one-electron operator called the *Fock* (or *Hartree-Fock*) *operator*, $\phi_i(1)$ is the ith spatial orbital and the eigenvalue ϵ_i is the *orbital energy* of spatial orbital ϕ_i. The Fock operator can be expressed as

$$\hat{F}(1) = \hat{h}^{core}(1) + \hat{v}^{HF}(1). \tag{1.16}$$

The term $\hat{h}^{core}(1)$, the *core-Hamiltonian operator*, describes the motion of a single electron in the field of all other nuclei. It consists of the operator for the kinetic energy of one electron, and potential-energy operators for the attraction between one electron and the nuclei:

$$\hat{h}^{core}(1) = -\frac{1}{2}\nabla_1^2 - \sum_{A=1}^{M}\frac{Z_A}{r_{1A}}. \tag{1.17}$$

The term $\hat{v}^{HF}(1)$, the *Hartree-Fock potential operator*, models electron-electron interaction. For a closed-shell system, it is defined as:

$$\hat{v}^{HF}(1) = \sum_{j=1}^{N/2}\left[2\hat{J}_j(1) - \hat{K}_j(1)\right] \tag{1.18}$$

where $\hat{J}_j(1)$ and $\hat{K}_j(1)$ are the Coulomb and exchange operators, respectively. The sum over j runs over the occupied spatial orbitals ϕ_i of the N-electron molecule. The Coulomb and exchange operators are defined as:

$$\hat{J}_j(1)\phi_i(1) \quad = \quad \left[\int \phi_j(2)\frac{1}{r_{12}}\phi_j(2)d\mathbf{r}_2\right]\phi_i(1) \tag{1.19}$$

$$\hat{K}_j(1)\phi_i(1) \ = \ \left[\int \phi_j^*(2)\frac{1}{r_{12}}\phi_i(2)d\mathbf{r}_2\right]\phi_j(1). \tag{1.20}$$

The Coulomb operator $\hat{J}_j(1)$ represents the average repulsion experienced by electron 1 due to the charge distribution associated with electron 2. The exchange operator $\hat{K}_j(1)$ represents of the electrostatic interaction of two overlapping charge densities. It arises from the antisymmetry requirement imposed on the total wavefunction Ψ with respect to electron interchange and has no classical interpretation.

The HF equation is a differential equation in which the Fock operator, $\hat{F}$, depends on its own eigenfunction. Solving the HF equation requires the knowledge of the wavefunction. However, the wavefunction is not known initially. In cases such as this, the problem must be solved iteratively. This is usually done by using an initial guess to the wavefunction to calculate the new wavefunction. Then, one takes the new wavefunction as the next guess. The procedure is repeated until the old and the new wavefunctions do not differ. This method is known as the *self-consistent field (SCF) method.* A practical solution to the problem of solving the HF differential equation was made possible by expanding the spatial orbitals ϕ_i as a linear combination of a known set of K one-electron basis function χ_ν [6]:

$$\phi_i(1) = \sum_{\nu=1}^{K} \chi_\nu(1)c_{\nu i}. \tag{1.21}$$

From hereon, the Greek letters μ, ν, λ and σ will be used to label the basis functions, and for simplicity, it is assumed that the basis functions are real. Substituting Eq. (1.21) into the HF equation (1.15) gives

$$\hat{F}(1)\sum_{\nu=1}^{K} \chi_\nu(1)c_{\nu i} = \epsilon_i \sum_{\nu=1}^{K} \chi_\nu(1)c_{\nu i} \tag{1.22}$$

Multiplying both sides of Eq. (1.22) from the left by a specific basis function, *e.g.*, $\chi_\mu(1)$, and integrating over the coordinates of electron 1, gives the Hartree-Fock-Roothaan-Hall (HFRH) equations [7]

$$\sum_{\nu=1}^{K} c_{\nu i}\left(\chi_\mu(1)\left|\hat{F}(1)\right|\chi_\nu(1)\right) = \epsilon_i \sum_{\nu=1}^{K} c_{\nu i}\left(\chi_\mu(1)|\chi_\nu(1)\right) \tag{1.23}$$

or in a more compact form

$$\sum_{\nu=1}^{K}(F_{\mu\nu} - \epsilon_i S_{\mu\nu})c_{\nu i} = 0 \qquad \mu = 1, 2, \ldots, K. \tag{1.24}$$

Eq. (1.24) can be conveniently represented in matrix notation as:

$$\mathbf{FC} = \mathbf{SC}\epsilon. \tag{1.25}$$

The **F** matrix contains the elements $F_{\mu\nu}$, consisting of the core, the Coulomb and the exchange contributions,

$$\begin{aligned} F_{\mu\nu} &= \sum_{\nu=1}^{K} c_{\nu i}\left(\chi_\mu(1)\left|\hat{h}^{core}(1)\right|\chi_\nu(1)\right) \\ &+ \sum_{\nu=1}^{K} c_{\nu i}\sum_{j=1}^{N/2}\left(\chi_\mu(1)\left|2\hat{J}_j(1) - \hat{K}_j(1)\right|\chi_\nu(1)\right) \end{aligned} \tag{1.26}$$

By further expanding the integrals involving the $\hat{J}$ and $\hat{K}$ operators using Eq. (1.21) into the corresponding integrals:

$$\begin{aligned} \left(\chi_\mu(1)\left|\hat{J}_j(1)\right|\chi_\nu(1)\right) &= (\chi_\mu(1)\chi_\nu(1)|\phi_j(2)\phi_j(2)) \\ &= \sum_{\lambda=1}^{K}\sum_{\sigma=1}^{K} c_{\lambda j}c_{\sigma j}\left(\chi_\mu(1)\chi_\nu(1)|\chi_\lambda(2)\chi_\sigma(2)\right), \end{aligned} \tag{1.27}$$

and

$$\begin{aligned} \left(\chi_\mu(1)\left|\hat{K}_j(1)\right|\chi_\nu(1)\right) &= (\chi_\mu(1)\phi_j(1)|\phi_j(2)\chi_\nu(2)) \\ &= \sum_{\lambda=1}^{K}\sum_{\sigma=1}^{K} c_{\lambda j}c_{\sigma j}\left(\chi_\mu(1)\chi_\sigma(1)|\chi_\lambda(2)\chi_\nu(2)\right), \end{aligned} \tag{1.28}$$

respectively, and introducing the charge density matrix, **P**, whose elements are defined as:

$$P_{\lambda\sigma} = 2\sum_{j=1}^{N/2} c_{\lambda j}c_{\sigma j}, \tag{1.29}$$

the expression for each element $F_{\mu\nu}$ of the Fock matrix for a closed-shell system of N electrons is:

$$F_{\mu\nu} = h_{\mu\nu}^{core} + \sum_{\lambda=1}^{K}\sum_{\sigma=1}^{K} P_{\lambda\sigma}[(\mu\nu|\lambda\sigma) - \frac{1}{2}(\mu\sigma|\lambda\nu)]. \tag{1.30}$$

The **S** matrix contains the elements $S_{\mu\nu}$, the overlap integrals between the basis function μ and ν

$$S_{\mu\nu} = (\mu|\nu). \tag{1.31}$$

C is a $K \times K$ square matrix of the expansion coefficients $c_{\nu i}$

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1K} \\ c_{21} & c_{22} & \cdots & c_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K1} & c_{K2} & \cdots & c_{KK} \end{pmatrix}, \tag{1.32}$$

and ϵ is a diagonal matrix of the orbital energies ϵ_i

$$\epsilon = \begin{pmatrix} \epsilon_1 & 0 & \cdots & 0 \\ 0 & \epsilon_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \epsilon_K \end{pmatrix}. \tag{1.33}$$

The solution to Eq. (1.25) requires finding the matrix **C** and ϵ. In solving the HFRH equation via the SCF procedure, an initial guess of the expansion coefficients $c_{\nu i}$ is made to form the Fock matrix. The Fock matrix is then diagonalized to obtain a new set of coefficients which is then used for forming the new Fock matrix. The procedure is repeated until there are no more changes between the old and the new set of coefficients within some given threshold. The converged set of coefficients constitutes the SCF solution.

In Eq. (1.30), the round bracket notation, $(\cdots|\cdots)$, has been adapted. In this notation, the orbitals, which are described by Eq. (1.21) and are identified by their subscripts, that are placed on the left refers to electron 1 and those on the right to electron 2, and that the complex conjugate function always being written first in the sequence. The other notation which is also worth mentioning is the angled bracket, $\langle\cdots|\cdots\rangle$. In this notation, the complex conjugate functions are written on the left-hand side and the real functions on the right. In this manuscript, either notation has been used whenever appropriate.

The generalized HF energy (or the total energy) of any closed shell system using N-occupied orbitals is:

$$E_{HF} = 2\sum_{i=1}^{N/2} \hat{h}_{ii}^{core} + \sum_{i=1}^{N/2}\sum_{j=1}^{N/2} [2(ii\,|\,jj) - (ij\,|\,ij)] + \sum_{A=1}^{M}\sum_{B>A}^{M} \frac{Z_A Z_B}{R_{AB}} \tag{1.34}$$

where

$$\hat{h}_{ii}^{core} \equiv \left(\phi_i^*(1)\left|\hat{h}^{core}(1)\right|\phi_i(1)\right) \tag{1.35}$$

$$(ii\,|\,jj) \equiv \left(\phi_i^*(1)\phi_i(1)\left|\frac{1}{r_{12}}\right|\phi_j^*(2)\phi_j(2)\right) = J_{ij} \tag{1.36}$$

$$(ij\,|\,ij) \equiv \left(\phi_i^*(1)\phi_j(1)\left|\frac{1}{r_{12}}\right|\phi_i^*(2)\phi_j(2)\right) = K_{ij} \tag{1.37}$$

in the round bracket notation. Equivalently, Eq. (1.34) in the angled bracket notation can be written as:

$$E_{HF} = 2\sum_{i=1}^{N/2} \hat{h}_{ii}^{core} + \sum_{i=1}^{N/2}\sum_{j=1}^{N/2} (2\langle ij\,|\,ij\rangle - \langle ij\,|\,ji\rangle) + \sum_{A=1}^{M}\sum_{B>A}^{M} \frac{Z_A Z_B}{R_{AB}} \tag{1.38}$$

where

$$\hat{h}_{ii}^{core} \equiv \left\langle\phi_i(1)\left|\hat{h}^{core}(1)\right|\phi_i(1)\right\rangle \tag{1.39}$$

$$\langle ij\,|\,ij\rangle \equiv \left\langle\phi_i(1)\phi_j(2)\left|\frac{1}{r_{12}}\right|\phi_i(1)\phi_j(2)\right\rangle = J_{ij} \tag{1.40}$$

$$\langle ij\,|\,ji\rangle \equiv \left\langle\phi_i(1)\phi_j(2)\left|\frac{1}{r_{12}}\right|\phi_j(1)\phi_i(2)\right\rangle = K_{ij}. \tag{1.41}$$

In a closed shell system, restricting each spatial orbitals to have two electrons, one with α and one with β spin, is known as the *Restricted Hartree-Fock* (RHF) method. For open shell systems, it is known as the *Restricted Open-shell Hartree-Fock* (ROHF) method.

1.1.2 Post Hartree-Fock Methods

The Hartree-Fock wavefunction is a good approximation to the many-body wavefunction for solving the Schrödinger equation. Although the HF method treats electron interaction in an average way, it is able to account for about 99% of the total energy of the system using a sufficiently large basis. In order to improve the HF approximation, instantaneous electron-electron interaction, *i.e. electron correlation*, must also be considered. The effect of the electrons being correlated is often described by the electron correlation energy, E_{corr}. E_{corr} is defined as the difference between the exact nonrelativistic energy of the system, (E_{NR}) and the HF energy, (E_{HF}), obtained by using a sufficiently large basis set,

$$E_{corr} \equiv E_{NR} - E_{HF} \tag{1.42}$$

A general method of improving the HF results is to consider more than one Slater determinant, Eq. (1.11), for the exact wavefunction. Such determinants may be constructed using the solutions of the HFRH equations. By solving the Roothaan-Hall equation of a closed-shell N-electron system using K basis functions, there are $2K$ spinorbitals of which N are occupied and $(2K - N)$ are virtual spinorbitals. From these $2K$ spinorbitals, a large number of determinants can be formed. Among these determinants, aside from the N lowest energy spin orbitals, $|\Phi_{HF}\rangle \equiv |\Phi_0\rangle$, are singly, $|\Phi_i^a\rangle$, doubly, $\left|\Phi_{ij}^{ab}\right\rangle$, etc., up to n-tuply excited determinants. These determinants can be used as a basis to expand the exact wavefunction:

$$|\Psi\rangle = c_0\,|\Phi_0\rangle + \sum_{ia} c_i^a\,|\Phi_i^a\rangle + \sum_{i<j;\ a<b} c_{ij}^{ab}\left|\Phi_{ij}^{ab}\right\rangle + \cdots \tag{1.43}$$

The determinants $|\Phi_0\rangle$, $|\Phi_i^a\rangle$, $\ldots$, are kept fixed, while the coefficients c_i are optimized. The best possible expansion coefficients c's are determined and the way they are calculated varies from one method to another. If the coefficient c_0 is large *i.e.*, close to 1, the HF wavefunction is a good representation of the true wavefunction.

There are several techniques for calculating electron correlation that have been reviewed [8, 9, 10]. Very commonly used among these methods are *configuration interaction* (CI), *coupled-cluster* (CC) and *perturbation theory* (PT). Only the perturbation theory is discussed here since the other methods are not used in this thesis. For a discussion of CI, see Ref. [2] and for a review of the CC method, see Ref. [11].

Estimating the electron correlation energy based on perturbation theory was one of the earliest post-HF procedures. The basic idea is that if there are two Hamiltonian operators that are fairly close to each other, for one of which the exact solution is known, then the difference between them is a small perturbation to the solvable Hamiltonian operator.

The *Møller-Plesset perturbation theory* (MPPT) [12] is commonly used in approximating the electron correlation energy. Recalling from Eq. (1.5), the true nonrelativistic electronic Hamiltonian ($\hat{H}_{el} \equiv \hat{H}$) for an N-electron system can be rewritten as:

$$\hat{H} = \sum_{i=1}^{N} \hat{h}^{core}(i) + \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{1}{r_{ij}} \tag{1.44}$$

which takes into account all electron correlation. If the HF wavefunction corresponds to the unperturbed Hamiltonian, $\hat{H}^{(0)}$,

$$\hat{H}^{(0)} = \sum_{i=1}^{N} \hat{F}(i) = \sum_{i=1}^{N} \left[\hat{h}^{core}(i) + v^{HF}(i)\right] \tag{1.45}$$

then electron correlation is seen merely as a perturbation

$$\hat{H}' = \hat{H} - \hat{H}^{(0)} = \sum_{i=1}^{N}\sum_{j>i}^{N} \frac{1}{r_{ij}} - \sum_{j=1}^{N}\sum_{m=1}^{N} \left[\hat{J}_m(j) - \hat{K}_m(j)\right]. \tag{1.46}$$

The second-order Møller-Plesset perturbation theory (MP2) is widely used in molecular calculations because it is relatively inexpensive and usually gives a reasonable portion of the correlation energy. For the ground state of a molecule, this energy is given as

$$E_0^{(2)} = \sum_{i<j}^{occ}\sum_{a<b}^{vir} \frac{\left[\langle ij \mid ab\rangle - \langle ij \mid ba\rangle\right]^2}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \tag{1.47}$$

where the shorthand notation similar to Eqs. (1.40) and (1.41) for the two electron integrals has been used. For third- and fourth-order MP energy correction formulas, see derivation by Krishnan and Pople [13].

1.1.3 Density Functional Theory

Post-Hartree-Fock methods are very good in approximating the exact solution to the Schrödinger equation. However, even for small molecular systems the calculations are very expensive and time consuming, especially if a very large basis set is used. This problem has been addressed by another popular and powerful method called the *density functional theory* (DFT). In the DFT, the system is described by the electron density ρ which depends only on three variables x, y and z. No matter how large the system is, the problem is always 3-dimensional and not $3N$-dimensional as in the HF-based methods. Change of the description of a system from a wave functional approach to density functional approach offers a tremendous reduction in computational effort needed to understand electronic properties of molecular systems.

Hohenberg and Kohn [14] proved in 1964 that for N interacting electrons moving in an external potential $v(\mathbf{r}_i)$, there is a universal functional $F[\rho(\mathbf{r})]$ of the ground-state electron density $\rho(\mathbf{r})$ that minimizes the energy functional

$$E[\rho(\mathbf{r})] = F[\rho(\mathbf{r})] + \int \rho(\mathbf{r})v(\mathbf{r}_i)d\mathbf{r} \tag{1.48}$$

The minimum value of the functional E is E_0, the ground-state electronic energy. The theory is exact only for a nondegenerate ground-state. For a discussion on degenerate ground-states, see Parr and Yang [15].

The major problem with the above theory is that neither the functional $F[\rho(\mathbf{r})]$ is known nor the ways of finding $\rho(\mathbf{r})$ without first finding the wavefunction. In 1965, Kohn and Sham [16] extended the applicability of the Hohenberg and Kohn theorem by devising a practical method for finding $\rho(\mathbf{r})$. This is known as the *Kohn-Sham* (KS) equation, which is very similar to the Hartree-Fock equation:

$$\hat{F}^{KS}\phi_i^{KS}(\mathbf{r}) = \epsilon_i^{KS}\phi_i^{KS}(\mathbf{r}) \tag{1.49}$$

where $\hat{F}^{KS}$ is the effective one-electron Kohn-Sham operator, $\phi_i^{KS}(\mathbf{r})$ are Kohn-Sham orbitals and ϵ_i are Kohn-Sham orbital energies. The interpretation of Kohn-Sham orbitals and orbital energies was discussed by Stowasser and Hoffmann [17]. A comparison of HF and KS determinants as wave functions was discussed by Bouř [18]. The Kohn-Sham orbitals, just like in the Hartree-Fock model, can be expanded as a linear combination of a set of K basis functions,

χ_α

$$\phi_i^{KS}(\mathbf{r}) = \sum_{\alpha=1}^{K} \chi_\alpha(\mathbf{r}) c_{\alpha i} \tag{1.50}$$

where the expansion coefficients $\{c_{\alpha i}\}$ are found iteratively.

In the Kohn-Sham formalism, the electron density of the system is calculated as:

$$\rho(\mathbf{r}) = 2\sum_{i=1}^{N/2} |\phi_i(\mathbf{r})|^2 . \tag{1.51}$$

The Kohn-Sham operator, $\hat{F}^{KS}(1)$, is defined as

$$\hat{F}^{KS}(1) = -\frac{1}{2}\nabla_1^2 - \sum_{A=1}^{M} \frac{Z_A}{r_{1A}} + 2\sum_{j=1}^{N/2} J_j + V_{XC}(\mathbf{r}) \tag{1.52}$$

where J_j is the Coulomb repulsion term

$$J_j(\mathbf{r}_1) = \int \frac{\rho(\mathbf{r}_2)}{r_{12}} dv_2 \tag{1.53}$$

and $V_{XC}(\mathbf{r})$ is the exchange-correlation potential term

$$V_{XC}(\mathbf{r}) = \frac{\delta E_{XC}[\rho(\mathbf{r})]}{\delta \rho(\mathbf{r})}. \tag{1.54}$$

The ground-state energy functional in Eq. (1.48) can then be expressed as:

$$\begin{aligned} E_0[\rho(\mathbf{r})] &= 2\sum_{i=1}^{N/2} \left\langle \phi_i(\mathbf{r}) \left| -\frac{1}{2}\nabla^2 \right| \phi_i(\mathbf{r}) \right\rangle - \sum_{A=1}^{M} \int \frac{Z_A \rho(\mathbf{r}_1)}{r_{1A}} d\mathbf{r}_1 \\ &+ \frac{1}{2} \int\int \frac{\rho(\mathbf{r}_1)\rho(\mathbf{r}_2)}{r_{12}} d\mathbf{r}_1 d\mathbf{r}_2 + E_{XC}[\rho(\mathbf{r})]. \end{aligned} \tag{1.55}$$

If the exact form of the exchange-correlation energy $E_{XC}[\rho(\mathbf{r})]$ is known, then the exact ground-state energy and density of the system can be calculated readily. However, $E_{XC}[\rho(\mathbf{r})]$ is not known. $E_{XC}[\rho(\mathbf{r})]$ is usually approximated by separating it into two parts, pure exchange and pure correlation functionals

$$E_{XC}[\rho(\mathbf{r})] = E_X[\rho(\mathbf{r})] + E_C[\rho(\mathbf{r})] \tag{1.56}$$

and the approximation is done within the local density approximation (LDA), generalised-gradient approximation (GGA) or a hybrid of the two.

In the local density approximation, $E_{XC}[\rho(\mathbf{r})]$ is expressed as

$$E_{XC}^{LDA}[\rho(\mathbf{r})] = \int \varepsilon_{XC}[\rho(\mathbf{r})]\rho(\mathbf{r})d\mathbf{r} \tag{1.57}$$

where $\varepsilon_{XC}[\rho(\mathbf{r})]$ is the exchange-correlation energy per particle (or energy density) of a homogeneous electron gas of density $\rho(\mathbf{r})$. The values of $\varepsilon_{XC}[\rho(\mathbf{r})]$ are based on Monte Carlo calculations by Ceperley and Alder [19] and interpolation procedures provided by Vosko, Wilk and Nusair (VWN) [20]. The term LDA in a more general case is called the *local spin density approximation* (LSDA). LSDA usually gives good results for systems with a fairly large homogeneous charge density $\rho(\mathbf{r})$. The main shortcoming of LSDA is when it is applied to a system with large inhomogeneity; it tends to overestimate electron correlation resulting in the over-binding of molecules, *i.e.*, too short bond lengths.

In order to improve the LSDA approach, $E_{XC}[\rho(\mathbf{r})]$ must not only depend on the electron density but also in its gradient, *i.e.*,

$$E_{XC}^{GGA}[\rho(\mathbf{r})] = \int \varepsilon_{XC}[\rho(\mathbf{r})]\rho(\mathbf{r})d\mathbf{r} + \int G_{XC}[\rho(\mathbf{r}), \nabla\rho(\mathbf{r})]d\mathbf{r} \tag{1.58}$$

This *non-local* method is called the *gradient corrected* or *generalised-gradient approximation* (GGA). The functional G_{XC} usually provides corrections to the limitations of E_{XC} in LSDA. However, there is no universal formula for obtaining G_{XC}. A large number of density functionals are available in the literature including functionals due to Becke (B or B88 [21], B95 [22], B97 [23]), Perdew and Wang (P86 [24], PW86 [25], PW91 [26]) and Lee, Yang and Parr (LYP [27]) to name a few.

Another method for formulating density functionals is by expressing the exchange-correlation energy by a suitable combination of LSDA, exact exchange and gradient correction terms. The resulting functional is often called a *hybrid functional*. An example is the *Becke 3-parameter functional* [28]:

$$E_{XC}^{B3} = E_{XC}^{LSDA} + a(E_X^{exact} - E_X^{LSDA}) + b\Delta E_X^B + c\Delta E_C^{GGA} \tag{1.59}$$

where a, b, and c are *semi-empirical coefficients* to be determined by an appropriate fit to experimental data, E_X^{exact} is the exact exchange energy, ΔE_X^B is Becke's correction to LSDA for exchange, and ΔE_C^{GGA} is the gradient correction for correlation. Commonly used hybrid functionals are the B3PW91 [28, 26] and B3LYP [28, 27].

1.2 Basis Sets

The solution to the wave functional and density functional method requires the use of spatial orbitals ϕ_i expanded as a linear combination of a set of one-electron basis functions χ,

$$\phi_i = \sum_{\alpha} \chi_\alpha c_{\alpha i}.$$

The use of a complete set of basis functions results in an exact solution. In practice, however, this is not possible because of the limitation in computer resources available. Instead, a finite set of basis functions is used. Basis sets must be properly chosen if accurate molecular results are desired. In choosing basis functions, one has to consider two basic requirements: first, the function must correctly describe the qualitative description of the orbitals, *i.e.*, it must exhibit correct asymptotic behaviour as $r \to 0$ and must not decay too rapidly at $r \to \infty$ and second, it should be relatively easy to evaluate computationally. There are two types of basis functions currently employed in molecular calculations, namely, *Slater-type functions* (STF) and *Gaussian-type functions* (GTF). The former satisfies the first requirement while the latter satisfies the second as we shall see later. This section discusses a brief introduction to basis functions and some of the terms and notation associated with them. For a more detailed and thorough reviews on basis sets, see Refs. [29, 30, 31].

1.2.1 Slater and Gaussian Type Functions

The basis function χ is normally expressed in spherical coordinates as

$$\chi(\zeta, n, l, m; r, \theta, \phi) = R_{nl}(\zeta; r) Y_{lm}(\theta, \phi) \tag{1.60}$$

where $R_{nl}(r)$ is the radial distribution function and $Y_{lm}(\theta, \phi)$ is the spherical harmonic function describing the shape of the orbital. The labels n, l and m are the principal, angular momentum and magnetic quantum numbers, respectively.

The Slater-type functions were initially used in atomic and molecular calculations due to their similarity to the atomic orbitals of the hydrogen atom. They are described in spherical coordinates as [29]:

$$\begin{aligned} \chi(\zeta, n, l, m; r, \theta, \phi) &= N r^{n-1} e^{-\zeta r} Y_{lm}(\theta, \phi) && (1.61) \\ N &= (2\zeta)^{n+\frac{1}{2}}[(2n)!]^{-\frac{1}{2}} && (1.62) \end{aligned}$$

where N is the normalization constant and ζ is the exponent that determines the extent of the radial function. Although STF has the correct behavior at $r \to 0$ and at $r \to \infty$, the function is not good enough for fast evaluation of the two-electron integrals. GTFs offer a solution. GTFs are described in two different forms – in spherical and in Cartesian coordinates [29]. The spherical Gaussian form is expressed as

$$\begin{aligned} \chi(\zeta, n, l, m; r, \theta, \phi) &= N r^{n-1} e^{-\zeta r^2} Y_{lm}(\theta, \phi) && (1.63) \\ N &= 2^{n+1}[(2n-1)!!]^{-\frac{1}{2}}(2\pi)^{-\frac{1}{4}}\zeta^{\frac{(2n+1)}{2}} && \\ n &= l+1, l+3, l+5, \dots && (1.64) \end{aligned}$$

The GTF in Cartesian Gaussian form, on the other hand, is

$$\chi(\zeta, l, m, n; x, y, z) = N e^{-\zeta r^2} x^l y^m z^n \qquad (1.65)$$

$$N = (2\pi)^{\frac{3}{4}}[(2l-1)!!(2m-1)!!(2n-1)!!]^{-\frac{1}{2}}\zeta^{\frac{[l+m+n+\frac{3}{2}]}{2}} \qquad (1.66)$$

where N is the normalization constant, x, y and z are Cartesian coordinates, and l, m and n are just integer exponents not to be mistaken for quantum numbers. Normally, the sum of l, m and n is defined as L, (*i.e.*, $L = x + y + z$) and is associated with the shape of the orbitals. $L = 0$ corresponds to s-type function, $L = 1$ to p-type functions, $L = 2$ to d-type functions, and so on [31]. In both forms of the GTF, ζ represents the orbital exponents. Although a GTF exhibits a zero slope instead of a cusp as $r \to 0$ and decays too fast as $r \to \infty$, evaluation of the two-electron integrals is rather fast and easy to implement computationally. Though a single GTF does not properly describe the orbital as the STF does, by combining a reasonable number of GTFs with different exponents and coeficients, an approximation to the shape of STF can be obtained.

Basis sets must be flexible enough to allow for the correct description of the electron distribution in a molecule. Increasing the flexibility, such as adding polarizing functions, and increasing the size of the basis functions, in general, improves the quality of the results obtained in molecular calculations. However, an increase in the size of the basis functions, like combining a large number of GTFs, also increases computational demands. One approach is to form linear combination of *primitive GTFs* with coefficients that are not allowed to change during the SCF procedure [2, 29]:

$$\chi_{\mu,L}^{cGTF} = \sum_{i=1}^{K} d_{\mu,i}^{L} \chi_{L}^{pGTF}(\zeta_i; r) \tag{1.67}$$

where $\chi_{\mu,L}^{cGTF}$ are called *contracted GTFs* (cGTFs), $d_{\mu,i}^{L}$ are refered to as contraction coefficients and L refers to the type of "orbital shapes" described above. $\chi_{\mu,L}^{cGTF}$ determines the number of basis function that is used in molecular calculations.

1.3 The Semi-empirical SCF MO and the NDDO Method

Hartree-Fock and post Hartree-Fock belong to methods widely known as *ab initio* methods. *Ab initio* calculations make no further approximation other than those of representing molecular orbitals by a linear combination of atomic orbitals (LCAO) and the use of the SCF procedure. The major limiting factor for *ab initio* calculations is the amount of computing resources available. This is so because the majority of computation time is spent calculating and manipulating integrals. For a system consisting of a few atoms and basis set of moderate size, the calculations, even at the Hartree-Fock level, can be very time consuming and often require very large amounts of computer memory and disk space for the storage of integrals. As an example, the cost of performing a Hartree-Fock calculation scales as N^4, where N is the number of basis functions used. Table 1.1 gives the typical scaling behavior of some of the common *ab initio* methods. Practically, *ab initio* calculations are limited to typically fewer than 50 atoms. For large molecular systems of biological interest, it would be impractical, if not impossible, to perform *ab initio* calculations unless further approximations are introduced.

Table 1.1: Scaling behavior of some *ab initio* methods.

HF	$\mathcal{O}(N^4)$
MP2	$\mathcal{O}(N^5)$
MP4	$\mathcal{O}(N^7)$
CISD	$\mathcal{O}(N^6)$
DFT	$\mathcal{O}(N^4)$

N = basis functions

The general formulation of the semi-empirical methods is based on the key components of the Hartree-Fock-Roothaan-Hall equations. For a closed-shell system, these equations are:

$$\mathbf{FC} = \mathbf{SC}\epsilon \tag{1.68}$$

$$F_{\mu\nu} = h_{\mu\nu}^{core} + \sum_{\lambda=1}^{K}\sum_{\sigma=1}^{K} P_{\lambda\sigma}[(\mu\nu|\lambda\sigma) - \frac{1}{2}(\mu\sigma|\lambda\nu)] \tag{1.69}$$

$$P_{\lambda\sigma} = 2\sum_{j=1}^{N/2} c_{\lambda j}c_{\sigma j} \tag{1.70}$$

$$\hat{h}_{\mu\nu}^{core} = \left(\mu\left|-\frac{1}{2}\nabla_1^2 - \sum_{A=1}^{M}\frac{Z_A}{r_{1A}}\right|\nu\right) \tag{1.71}$$

where μ, ν, λ, and σ denote AOs.

Semi-empirical methods try to reduce the computing time by simplifying the Hartree-Fock-Roothaan approach. This is accomplished by: (1) treating explicitly only the valence electrons in the construction of the Ψ_0; (2) considering only a minimal atomic basis set for calculations (*e.g.*, H atoms are described by a $1s$ function, the elements Li–F by $\{2s,2p\}$ functions, the elements Na–Cl by $\{3s, 3p\}$ functions, and so on), and; (3) neglecting a large part of the interaction, particularly those that involve the two-electron integrals, for example, all integrals involving AOs centered at more than two centers are neglected while some integrals are replaced by parameterized functions. These simplifications are responsible for a tremendous reduction in computational cost in semi-empirical methods over Hartree-Fock, post-Hartree-Fock and density functional methods. In fact, semi-empirical methods formally scale as N^2.

The fundamental approximation that is used in practically all modern semi-empirical SCF MO methods is the zero differential overlap (ZDO). In the ZDO approximation, it is assumed that the overlap between pairs of different atomic orbitals is zero:

$$(\mu_A|\nu_B) = 0, \quad \mu_A \neq \nu_B. \tag{1.72}$$

Eq. (1.72) has several important consequences, namely, (1) the overlap matrix **S** reduces to a unit matrix, and (2) only the "diagonal" terms of type $(\mu\mu|\lambda\lambda)$ remain from all the two-electron integrals $(\mu\nu|\lambda\sigma)$.

The introduction of ZDO approximation led to the various classifications of approximate semi-empirical SCF MO scheme [32] according to the degree and range of additional approximations incorporated into their HFRH equations. These schemes are CNDO (*Complete Neglect of Differential Overlap*) [32, 33, 34], INDO (*Intermediate Neglect of Differential Overlap*) [35], and NDDO (*Neglect of Diatomic Differential Overlap*) [32], respectively. More focus is given to the NDDO scheme since this is the basis for AM1 and PM3 methods that are used in this thesis.

The NDDO method was originally proposed Pople, Santry and Segal [32] and later developed by Pople and Beveridge [36]. In the NDDO method, the ZDO approximation (Eq. (1.72)) is used for all overlap integrals, i.e.,

$$S_{\mu_A\nu_B} = \delta_{\mu_A\nu_B} = \delta_{\mu\nu}\delta_{AB}, \tag{1.73}$$

and for it to be valid for treating two-electron integrals, the AOs ascribed to a given electron must belong to different atoms:

$$(\mu_A\nu_B|\lambda_C\sigma_D) = (\mu_A\nu_A|\lambda_C\sigma_C)\delta_{AB}\delta_{CD}. \tag{1.74}$$

The NDDO method assumes that the differential overlap of atomic orbitals on different atoms is neglected in both overlap and two-electron integrals, including those two-electron integrals involving three- and four-centers. All the two-center two-electron integrals of the form $(\mu\nu|\lambda\sigma)$ are retained, provided that μ and ν are on the same atom and λ and σ are also on the same atom. With this, the NDDO Fock matrix elements become:

$$\begin{aligned} F_{\mu\mu} &= \hat{h}^{core}_{\mu\mu} + \sum_{\lambda}^{A}\sum_{\sigma}^{A}\left[P_{\lambda\sigma}(\mu\mu|\lambda\sigma) - \frac{1}{2}P_{\lambda\sigma}(\mu\lambda|\mu\sigma)\right] \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\mu|\lambda\sigma) \qquad (1.75) \\ F_{\mu\nu} &= \hat{h}^{core}_{\mu\nu} + \sum_{\lambda}^{A}\sum_{\sigma}^{A}\left[P_{\lambda\sigma}(\mu\nu|\lambda\sigma) - \frac{1}{2}P_{\lambda\sigma}(\mu\lambda|\nu\sigma)\right] \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\nu|\lambda\sigma); \quad \mu \text{ and } \nu \text{ both on A} \qquad (1.76) \\ F_{\mu\nu} &= \hat{h}^{core}_{\mu\nu} - \frac{1}{2}\sum_{\lambda}^{B}\sum_{\sigma}^{A} P_{\lambda\sigma}(\mu\sigma|\nu\lambda); \quad \mu \text{ on A and } \nu \text{ on B}. \qquad (1.77) \end{aligned}$$

Using an {s, p} basis sets, Eqs. (1.75)–(1.77) can be simplified to

$$\begin{aligned} F_{\mu\mu} &= \hat{h}^{core}_{\mu\mu} + \sum_{\nu}^{A}\left[P_{\nu\nu}(\mu\mu|\nu\nu) - \frac{1}{2}P_{\nu\nu}(\mu\nu|\mu\nu)\right] \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\mu|\lambda\sigma) \qquad (1.78) \\ F_{\mu\nu} &= \hat{h}^{core}_{\mu\nu} + \frac{3}{2}P_{\mu\nu}(\mu\nu|\mu\nu) - \frac{1}{2}P_{\mu\nu}(\mu\mu|\nu\nu) \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\nu|\lambda\sigma) \qquad (1.79) \end{aligned}$$

Modern semi-empirical methods, such as MNDO, AM1 and PM3, are based on the NDDO model and have been implemented and included in many popular computer programs such as GAMESS-US [37], GAMESS-UK [38], Gaussian [39], DYNAMO [40], and MOPAC2007 [41] to name a few. Recent semi-empirical reparameterizations have also been introduced such as *Recife Model 1* (RM1) [42] and PM6 [43].

1.3.1 MNDO Method

The *Modified Neglect of Diatomic Overlap* (MNDO) method was introduced by Dewar and Thiel [44]. MNDO was originally developed for the first-row elements (H, C, N, and O) [45] and later extended by Thiel and Voityuk to halogens, the second-row elements and the transition metals after inclusion of d functions (MNDO-d) [46, 47, 48, 49]. The Fock matrix elements of MNDO are very similar to that of the NDDO with the introduction of new terms for the $\hat{h}^{core}$, as shown in the following equations:

$$\begin{aligned} F_{\mu\mu} &= \hat{h}^{core}_{\mu\mu} + \sum_{\nu}^{A}\left[P_{\nu\nu}(\mu\mu|\nu\nu) - \frac{1}{2}P_{\nu\nu}(\mu\nu|\mu\nu)\right] \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\mu|\lambda\sigma) \qquad (1.80) \\ \text{where} \quad & \hat{h}^{core}_{\mu\mu} = U_{\mu\mu} - \sum_{B\neq A} V_{\mu\mu B} \qquad (1.81) \end{aligned}$$

$$\begin{aligned} F_{\mu\nu} &= \hat{h}^{core}_{\mu\nu} + \frac{3}{2}P_{\mu\nu}(\mu\nu|\mu\nu) - \frac{1}{2}P_{\mu\nu}(\mu\mu|\nu\nu) \\ &+ \sum_{B\neq A}\sum_{\lambda}^{B}\sum_{\sigma}^{B} P_{\lambda\sigma}(\mu\nu|\lambda\sigma); \quad \mu \text{ and } \nu \text{ both on A} \qquad (1.82) \\ \text{where} \quad & \hat{h}^{core}_{\mu\nu} = -\sum_{B\neq A} V_{\mu\nu B} \qquad (1.83) \end{aligned}$$

$$\begin{aligned} F_{\mu\nu} &= \hat{h}^{core}_{\mu\nu} - \frac{1}{2}\sum_{\lambda}^{B}\sum_{\sigma}^{A} P_{\lambda\sigma}(\mu\sigma|\nu\lambda); \quad \mu \text{ on A and } \nu \text{ on B} \qquad (1.84) \\ \text{where} \quad & \hat{h}^{core}_{\mu\nu} = \frac{1}{2}S_{\mu\nu}(\beta_\mu + \beta_\nu) \qquad (1.85) \end{aligned}$$

The $U_{\mu\mu}$ in Eq. (1.81) represents the one-center, one-electron integrals. The $V_{\mu\mu B}$ and $V_{\mu\nu B}$ in Eqs. (1.81) and (1.83) are the two-center, one-electron attraction terms between an electron distribution $\phi_\mu\phi_\mu$ or $\phi_\mu\phi_\nu$, respectively, on atom A and the core of atom B, and are expressed as follows:

$$V_{\mu\mu B} = -Z_B(\mu_A\mu_A|s_Bs_B) \qquad (1.86)$$

$$V_{\mu\nu B} = -Z_B(\mu_A\nu_A|s_Bs_B) \qquad (1.87)$$

The $\hat{h}_{\mu\nu}^{core}$ in Eq. (1.85), which represents the two-center, one-electron core resonance integrals, depends on the overlap $S_{\mu\nu}$ and the parameters β_μ and β_ν. In MNDO, the core-core repulsion terms are represented as either:

$$E_{AB}^{MNDO} = Z_A Z_B (s_A s_A | s_B s_B)(1 + e^{-\alpha_A R_{AB}} + e^{-\alpha_B R_{AB}}), \qquad (1.88)$$

or

$$E_{XH}^{MNDO} = Z_X Z_H (s_X s_X | s_H s_H)(1 + R_{XH} e^{-\alpha_X R_{XH}/R_{AB}} + e^{-\alpha_H R_{XH}}) \qquad (1.89)$$

if $O-H$ and $N-H$ bonds are present. The Z_A and Z_B correspond to the core charges, R_{AB} is the internuclear separation, and α_A and α_B in the Gaussian are some adjustable parameters. The MNDO method is extended and used as the basis for semi-emprical methods such as AM1 and PM3.

1.3.2 The AM1 and PM3 Methods

The AM1 method or the Austin Model 1 was developed by Dewar and co-workers [50, 51] in which many of its parameters were obtained by applying chemical knowledge. The PM3 or MNDO-*Parameterization Method 3*, on the other hand, was developed by Stewart [52, 53] in which its parameters were derived using an automated parameterization procedure.

AM1 and PM3 were developed to eliminate problems associated with MNDO. MNDO tends to overestimate repulsions between atoms separated by distances that are approximately equal to the sum of their van der Waals radii. The correction was done by adding to the MNDO core-core repulsion function both attractive and repulsive Gaussian functions. The attractive Gaussian functions were centered in the region where the repulsions were too large. The repulsive Gaussian functions, on the other hand, were centered at smaller internuclear separations. Both AM1 and PM3 uses the same functional form (Eq. (3.1)):

$$\begin{aligned} E_{AB}^{AM1,PM3} &= E_{AB}^{MNDO} + \frac{Z_A Z_B}{R_{AB}} \\ &\times \left(\sum_i a_{iA} e^{-b_{iA}(R_{AB}-c_{iA})^2} + \sum_j a_{jB} e^{-b_{jB}(R_{AB}-c_{jB})^2} \right), \end{aligned} \qquad (1.90)$$

where a, b, and c are adjustable parameters that describe Gaussian functions centered at various distances c. These parameters tend to modify the potential of mean force between the two atoms. The AM1 has between two and four Gaussians per atom, while PM3 has two.

1.3.3 The Semi-empirical Parameters

As previously mentioned, the ability of the semi-empirical methods to treat large molecular systems rely on the simplification of the HFRH equation by introducing empirical parameters. These parameters are either experimentally derived fixed parameters or adjustable parameters obtained by optimization procedures. Table 1.2 summarizes the parameters and their corresponding units used in semi-empirical methods MNDO, AM1, and PM3.

Table 1.2: Semi-empirical parameters for MNDO, AM1, and PM3.

Parameter	Units	Description
U_{ss}, U_{pp}	eV	s and p atomic orbital one-electron one-center integral terms
β_s, β_p	eV	s and p atomic orbital one-electron two-center resonance integral terms
ζ_s, ζ_p	bohr^{-1}	s- and p-type Slater atomic orbital exponents
α_A	Å^{-1}	atom A core-core repulsion term
G_{ss}	eV	$s-s$ atomic orbital one-center two-electron repulsion integral term
G_{sp}	eV	$s-p$ atomic orbital one-center two-electron repulsion integral term
G_{pp}	eV	$p-p$ atomic orbital one-center two-electron repulsion integral term
G_{p2}	eV	$p-p'$ atomic orbital one-center two-electron repulsion integral term
H_{sp}	eV	$s-p$ atomic orbital one-center two-electron exchange integral term
a_{nA}	–	Gaussian multiplier for the nth Gaussian of atom A
b_{nA}	Å^{-2}	Gaussian exponent multiplier for the nth Gaussian of atom A
c_{nA}	Å	radius of center of the nth Gaussian of atom A

1.4 Classical Molecular Mechanics Method

Molecular mechanics (MM) or empirical force field method is the most commonly used type of calculation method when dealing with large molecular systems, since it offers several orders of magnitude faster than quantum mechanical ones. Molecular mechanics methods rely on two fundamental assumptions: (1) that the total potential energy of a molecular structure can be expressed as a sum of contributions from many interaction types, such as bond stretching, angle bending, bond rotation, and non-bonded interactions; and (2) that the force field and its parameters are transferable to other molecules. Transferability simply means that the same set of parameters can be used or "transfered" to model a series of related compounds without having to redefine a new set of parameters for each individual molecule.

One simple functional form for a force field that can be used to model a single molecule or a collection of molecules and/or atoms is:

$$\begin{aligned} V(\mathbf{r}^N) &= \sum_{bonds} \frac{1}{2} k_b (r - r_0)^2 + \sum_{angles} \frac{1}{2} k_a (\theta - \theta_0)^2 \\ &+ \sum_{torsions} \frac{1}{2} V_n [1 + cos(n\omega - \gamma)] \\ &+ \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} \left\{ 4\epsilon_{i,j} \left[\left(\frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}} \right\}. \qquad (1.91) \end{aligned}$$

$V(\mathbf{r}^N)$ denotes the potential energy that is a function of the positions ($\mathbf{r}$) of N atoms. In Eq. (1.91), the first term represents the energy of interaction between covalently bonded atoms modelled by a harmonic potential that gives the increase in energy as the bond length r deviates from equilibrium bond length r_0. The second term represents the energy due to the valence angles (formed by three atoms, say A–B–C, in which both A and C are bonded to B) in the molecules, again modelled by a harmonic potential. k_b and k_a are force constants. The third term represents the energy for twisting a bond. Torsional potentials are almost always expressed as a cosine series, where V_n is the barrier height to rotation. The fourth term represents the non-bonded energy between all pairs of atoms (i and j) that are in different molecules or that are in the same molecule but separated by at least three bonds. The non-bonded energy term given in Eq. (1.91) consists of van der Waals interactions, which are represented by a Lennard-Jones 12-6 potentials, and electrostatic interactions, which are represented by interactions of point charges q_i and q_j.

One of the fundamental concepts for the majority of force field that has been developed is the *atom type*. The atom type is used for determining what parameters to apply for all interactions needed in the calculations. The atom type usually contains information such as bond lengths, bond angles, dihedral angles, bond orders, hybridization state of an atom, etc. There are many MM force fields currently in use. These force fields may or may not share the same definition for an atom type. Some of the available force fields are the MMn force fields of Allinger and co-workers (MM2 [54], MM3 [55, 56, 57, 58, 59], and MM4 [60, 61, 62, 63, 64]), the AMBER force field [65, 66], the CHARMm force field [67], the Dreiding force field [68], the UFF (Universal Force Field) [69, 70, 71], the CFF (Consistent Force Field) [72, 73, 74], the CFF93 (Consistent Force Field 93) [75, 76], the MMFF94 (Merck molecular force field) [77, 78, 79, 80, 81, 82, 83], and the OPLS (Optimized Potential for Liquid Simulation) force field [84, 85]. There are also force fields designed for the simulation of liquid water, such as TIP3P and TIP4P [86] and SPC [87]. The discussion of the differences among these force fields and their parameterizations is beyond the scope of this thesis.

1.5 Hybrid Quantum Mechanics and Molecular Mechanics Methods

Molecular modeling through computer simulations is one of the techniques used for searching and identifying possible anti-cancer and chemotherapy drugs. The technique requires a combination of accurate molecular models and powerful computers. Modeling biomolecular systems and their interactions, e.g., protein-ligand interactions, requires the inclusion of a large number of atoms in the model. The system is often divided into two subsystems: (1) small molecules of importance that are interacting with the protein and are possibly involved in a chemical reaction, plus atoms or molecules in the vicinity of these important molecules, and (2) the remaining parts of the system that act as "spectator" solvent. The first part is treated quantum mechanically and the second with classical mechanics. This technique is known as the *hybrid quantum mechanical and molecular mechanical* (QM/MM) method [88].

In the QM/MM method, the effective Hamiltonian, $\hat{H}_{eff}$, is dependent on the position vector of the QM electrons, r_i, the position vector of MM atoms, R_M, where M denotes an MM atom, and the position vector of the QM nuclei, R_A, where A denotes a QM atom. The $\hat{H}_{eff}$ operates on the wavefunction of the system:

$$\hat{H}_{eff}\Psi(r_i, R_M, R_A) = E_{eff}\Psi(r_i, R_M, R_A) \tag{1.92}$$

and is composed of three parts – one that describes the QM region, one that describes the MM region, and another that describes the interaction between the QM and MM regions:

$$\hat{H}_{eff} = \hat{H}_{MM} + \hat{H}_{QM} + \hat{H}_{QM/MM}. \tag{1.93}$$

The effective energy of the system can be written as follows:

$$\begin{aligned} E_{eff} &= E_{MM} + \left\langle \Psi | \hat{H}_{QM} + \hat{H}_{QM/MM} | \Psi \right\rangle && (1.94) \\ &= E_{MM} + E_{QM} + E_{QM/MM}. && (1.95) \end{aligned}$$

In Eq. (1.94), the MM term is not included in the integral since it is independent of the distribution of electrons. E_{MM} is calculated using an expression implemented in the force field like Eq. (1.91), and E_{QM} is the energy of the quantum subsystem:

$$E_{QM} = \left\langle \Psi | \hat{H}_{QM} | \Psi \right\rangle. \tag{1.96}$$

$E_{QM/MM}$ is the interaction energy between the quantum and classical subsystems. $E_{QM/MM}$ is calculated as

$$\begin{aligned} E_{QM/MM} &= \left\langle \Psi | \hat{H}_{QM/MM} | \Psi \right\rangle && (1.97) \\ &= \left\langle \Psi | \hat{V}_{elect} + \hat{V}_{vdW} | \Psi \right\rangle. && (1.98) \end{aligned}$$

The $\hat{V}_{elect}$ consists of two electrostatic terms, one in which the MM atoms interact with the QM electrons (at r_i) and the other in which the MM atoms (at R_M) interact with the QM nuclei (at R_A). The $\hat{V}_{elect}$, in atomic units, can be expressed as:

$$\hat{V}_{elect} = -\sum_{i,M} \frac{q_M}{r_{iM}} + \sum_{A,M} \frac{Z_A q_M}{R_{AM}} \tag{1.99}$$

where the sum runs to all M molecular mechanics atoms with charge q_M, all A nuclei of the quantum subsystems with charge Z_A, and all electrons i. The r_{iM} and R_{AM} are the interatomic distances. The V_{vdW} in Eq. (1.98), on the other hand, is the van der Waals interaction to account for the fact that the classical atoms are not point charges but consist of an attractive dispersion interaction with any other atom and a short-range repulsive interaction. This is necessary because some MM atoms do not have charges and therefore are invisible to the QM atoms. In addition, the $\hat{V}_{vdW}$ on the MM atoms often provide the only difference in the interactions of one atom type against another. The $\hat{V}_{vdW}$ has empirical values independent from the actual electronic structure of the interacting atoms, and typically has a form of:

$$\hat{V}_{vdW} = \sum_{A,M} \left\{ \frac{B_{AM}}{R_{AM}^{12}} - \frac{C_{AM}}{R_{AM}^{6}} \right\} \tag{1.100}$$

where B and C are some parameters.

1.5.1 The Link Atom Approach

There are no simple rules on how to partition a large biomolecular system into QM and MM regions. In studying protein-ligand interactions, for example, the ligand, which is normally a small molecule, is usually treated quantum mechanically while the rest of the system is treated classically or molecular mechanically. If a part of the system is directly involved in a chemical reaction, again that subsystem would have to be treated quantum mechanically. For example, in enzyme reactions, the ligand and some of the residues of the protein that play important catalytic roles would have to be treated quantum mechanically. Partitioning this kind of system into a quantum and classical subsystem would normally require cutting through bonds leading to the classification of atoms as QM on one side and MM on the other. Since the electrons of the MM atoms are not treated explicitly, such cutting of bonds linking QM and MM atoms would lead to creation of molecular fragments with dangling bonds. These dangling bonds would have to be saturated in order to perform QM calculations on the fragments. Therefore, it is necessary to devise a scheme in which the QM electron density along the QM/MM bonds can terminate satisfactorily. Strategies such as the *capping-atom*, the *frozen- or localized-orbital*, and the *link-atom* methods are commonly employed in these situations. For completeness, the capping-atom and the frozen- or localized-orbital methods are described very briefly. More details are given for the link-atom method since this is the method used in this thesis.

In the *capping-atom* method [89], atoms of fictitious elemental type are used to "cap" the bonds between atoms in the QM and MM regions. The capping atoms can be hydrogen-like (with one valence electron), or halogen-like (with seven valence electrons). The bonds between the capping atom and the MM atoms to which it is bound are reproduced using MM terms. In the *frozen-* or *localized-orbital* method, localized orbitals on the QM atom at the boundary of the MM region are used. The orbitals and their corresponding electron which points towards the MM atoms are then frozen. (See Refs. [90, 91, 92] for more details.)

The *link-atom* method was introduced by Singh and Kollman [93] and implemented by Field [94] in semi-empirical calculations in which the link atoms are represented and treated exactly like the QM hydrogen atoms. The link-atom scheme uses the same form of Hamiltonian that was developed for the semi-empirical methods for the interactions between the QM and MM atoms. The one-electron integrals between the basis functions μ and ν on the quantum atom A and the partial charge q_M on the MM atom is represented as:

$$I_{\mu\nu} = -q_M(\mu_A\nu_A|s_M s_M) \tag{1.101}$$

where s_M is an s-type orbital on an MM atom. The MNDO core-core energy of interaction between the point charges q_M of the classical subsystem and the atomic core charges Q_A of the quantum subsystem has the form:

$$\begin{aligned} E^{MNDO}_{QM/MM} &= \sum Q_A q_M(s_A s_A|s_M s_M) \\ &+ |Q_A q_M|(s_A s_A|s_M s_M)\{e^{-\alpha_A R_{AM}} + e^{-\alpha_M R_{AM}}\}. \end{aligned} \tag{1.102}$$

The $E_{QM/MM}^{AM1,PM3}$ is calculated by adding additional terms to $E_{QM/MM}^{MNDO}$:

$$\begin{aligned} E_{QM/MM}^{AM1,PM3} = E_{QM/MM}^{MNDO} &+ \sum_i \frac{Q_A q_M}{R_{AM}} a_i^A e^{-b_i^A (R_{AM} - c_i^A)^2} \\ &+ \sum_j \frac{Q_A q_M}{R_{AM}} a_j^M e^{-b_j^M (R_{AM} - c_j^M)^2}. \end{aligned} \tag{1.103}$$

The optimum values for the parameter, α_M in Eq. (1.102), and the parameters a_j^M, b_j^M and c_j^M in Eq. (1.103), were obtained by calculating interaction curves for charge/ion systems and comparing them with *ab initio* results [94]. It was found that the best value for α_M is 5.0 and those for a_j^M, b_j^M and c_j^M are all 0.0. In the link-atom scheme, both the QM/MM van der Waals and electrostatic interactions are calculated. The link atoms are invisible from the MM atoms, have no interactions with the MM atoms, and appear only in the QM portion of the calculation. However, the link atoms do feel the forces from the other QM atoms in the calculation.

1.6 Derivatives of the Energy Function

The energy of a system and their corresponding expressions for both the quantum mechanics and the molecular mechanics methods has been the focus of the discussion. In many applications, however, it is important to know the values of the derivatives of the energy function E with respect to the positions of the atoms, $\mathbf{r}_i$. The derivatives can provide very useful information, especially when seeking the minimum energy configuration of the system.

The first derivative of the energy or the *gradient*, $\mathbf{g}_i$,

$$\mathbf{g}_i = \frac{\partial E}{\partial \mathbf{r}_i} = \begin{pmatrix} \frac{\partial E}{\partial x_i} \\ \frac{\partial E}{\partial y_i} \\ \frac{\partial E}{\partial z_i} \end{pmatrix}, \quad i = 1, \ldots, N \text{ atoms} \tag{1.104}$$

is useful in guiding the system towards achieving lower energy configuration. The direction of the gradient points towards the location of the minimum and the magnitude of the gradient indicates the steepness of the local slope. The second derivative of the energy with respect to the coordinates of two atoms,

i and j, or the *Hessian*, $\mathbf{h}_{ij}$, is given as

$$\mathbf{h}_{ij} = \frac{\partial^2 E}{\partial \mathbf{r}_i \partial \mathbf{r}_j} = \begin{pmatrix} \frac{\partial^2 E}{\partial x_i \partial x_j} & \frac{\partial^2 E}{\partial x_i \partial y_j} & \frac{\partial^2 E}{\partial x_i \partial z_j} \\ \frac{\partial^2 E}{\partial y_i \partial x_j} & \frac{\partial^2 E}{\partial y_i \partial y_j} & \frac{\partial^2 E}{\partial y_i \partial z_j} \\ \frac{\partial^2 E}{\partial z_i \partial x_j} & \frac{\partial^2 E}{\partial z_i \partial y_j} & \frac{\partial^2 E}{\partial z_i \partial z_j} \end{pmatrix}. \tag{1.105}$$

The Hessian indicates the curvature of the energy function E which is used to predict where E will change direction.

The derivatives of E can be calculated either *analytically* or *numerically*. Analytical derivatives are calculated using explicit formulae for the derivative obtained by direct differentiation of the energy function. Numerical derivatives, on the other hand, are calculated using finite difference approximations, such as the *central-step finite difference method*, where for example, the gradient in the x-direction is approximated as:

$$\frac{\partial E(x)}{\partial x} \approx \frac{E(x + \Delta x) - E(x - \Delta x)}{2\Delta x} \tag{1.106}$$

in which Δx is a small change in the value of x. The use of analytical derivatives is preferable over the use of numerical derivatives because they are accurate to machine precision and can be calculated more quickly. The accuracy of numerical derivatives depend to a large extent on the values of the finite steps, e.g. Δx taken in the numerical algorithm. Also, the calculation of numerical derivatives can be very expensive, especially when a large number of derivatives needs to be calculated.

1.7 Energy Minimization and Simulated Annealing

Energy minimization techniques can be classified according to the amount of information about the energy function that is being used, as represented by the highest derivative used in the method. Methods that use only the value of the energy function and no derivatives are referred to as *zero-order* methods. Zero-order methods are crude and are rarely used for biomolecular systems. These methods, however, are used for systems with very simple energy surfaces.

First-order methods use the slope or first derivative and are often used in biomolecular simulations. These methods iterate using Eq. (1.107)

$$\mathbf{r}_k = \mathbf{r}_{k-1} + \lambda_k \mathbf{s}_k \tag{1.107}$$

by gradually changing the coordinates of the atoms as they move closer to the minimum. In Eq. (1.107), $\mathbf{r}_k$ is the new position at step k, $\mathbf{r}_{k-1}$ is the position at the previous step $k-1$, λ_k is the step size to be taken at step k, and $\mathbf{s}_k$ is direction or gradient unit vector of that step. The two frequently used algorithms belonging to this class are the method of *steepest descents* and the *conjugate gradient.* These two algorithms differ in the way the step size is chosen and the direction of the step.

Second-order methods use not only the slope of the energy function but also the curvature or second derivative to locate a minimum. The algorithms belonging to this class are the *Newton-Raphson* method and its variants.

Energy minimization techniques often find the local minimum closest to the initial geometry and not necessarily the global minimum. *Simulated annealing* is a method that is often used to find the global minimum on a complex potential energy surface. It involves heating up the system to a high temperature and then enabling the system wander around the energy landscape jumping out from one local minimum to another. This is then followed by gradual cooling causing the system to settle down to a lower local minimum further away from the initial conformation. Simulated annealing is often implemented in molecular dynamics simulations (Section 1.8) since the temperature can be easily manipulated in molecular dynamics.

1.8 Molecular Dynamics

The classical Hamiltonian, which defines the total energy of the system, consists of the sum of the kinetic and potential terms and can be written as:

$$\hat{H}(\mathbf{p}_i, \mathbf{r}_i) = \sum_{i=1}^{N} \frac{\mathbf{p}_i}{2m_i} + \sum_{i>j,j=1}^{N} V(\mathbf{r}_{ij}) \tag{1.108}$$

where m_i, $\mathbf{r}_i$ and $\mathbf{p}_i$ are the mass, position vector and momentum of the ith particle, respectively. The $V(\mathbf{r}_{ij})$ corresponds to the interaction energy. In molecular mechanics, $V(\mathbf{r}_{ij})$ is equivalent to the empirical energy function. In studying the dynamics of the system, or for performing molecular dynamics simulations, the Hamilton equations of motion:

$$\dot{\mathbf{r}}_i = \mathbf{v}_i = \frac{\partial \hat{H}}{\partial \mathbf{p}_i} = \frac{\mathbf{p}_i}{m_i} \tag{1.109}$$

$$\dot{\mathbf{p}}_i = -\frac{\partial \hat{H}}{\partial \mathbf{r}_i} = \mathbf{f}_i \tag{1.110}$$

or the Newton equations of motion:

$$m_i \ddot{\mathbf{r}}_i = m_i \mathbf{a}_i = \mathbf{f}_i \tag{1.111}$$

where $\mathbf{v}_i$ is the velocity, $\mathbf{a}_i$ is the acceleration, and $\mathbf{f}_i$ is the force, must be solved for each particle. Eqs. (1.109–1.111) are ordinary differential equations where a large number of integration algorithms exists, some of which are described next. The subscripts i have been dropped in the succeeding discussions for convenience.

1.8.1 The Integration Algorithms

Integration algorithms for molecular dynamics assume that the position and dynamic properties, such as velocities and accelerations, can be approximated as Taylor series expansions:

$$\mathbf{r}(t+\Delta t) = \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \frac{1}{6}\mathbf{b}(t)\Delta t^3 + \frac{1}{24}\mathbf{c}(t)\Delta t^4 + \cdots \tag{1.112}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t + \frac{1}{2}\mathbf{b}(t)\Delta t^2 + \frac{1}{6}\mathbf{c}(t)\Delta t^3 + \cdots \tag{1.113}$$

$$\mathbf{a}(t+\Delta t) = \mathbf{a}(t) + \mathbf{b}(t)\Delta t + \frac{1}{2}\mathbf{c}(t)\Delta t^2 + \cdots \tag{1.114}$$

$$\mathbf{b}(t+\Delta t) = \mathbf{b}(t) + \mathbf{c}(t)\Delta t + \cdots \tag{1.115}$$

where $\mathbf{v}$ is the first derivative of the positions with respect to time or the velocity, $\mathbf{a}$ is the second derivative or the acceleration, $\mathbf{b}$ is the third derivative, $\mathbf{c}$ is the fourth derivative, and so on.

The *Verlet algorithm* [95] is one of the most widely used integration scheme for solving the equations of motion in molecular dynamic simulations. It is based on the Taylor series expansion given above where it uses the positions $\mathbf{r}(t)$ and the accelerations $\mathbf{a}(t)$ at time t, and the positions from previous steps $\mathbf{r}(t-\Delta t)$ to calculate the new positions $\mathbf{r}(t+\Delta t)$ at $t+\Delta t$:

$$\mathbf{r}(t+\Delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\Delta t) + \mathbf{a}(t)\Delta t^2. \tag{1.116}$$

Eq. (1.116) is obtained by adding the expressions for the positions at time $t+\Delta t$ and $t-\Delta t$, then rearranging to obtain the expression for $\mathbf{r}(t+\Delta t)$:

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) && (1.117)\\
\mathbf{r}(t-\Delta t) &= \mathbf{r}(t) - \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 - \mathcal{O}(\Delta t^3). && (1.118)
\end{aligned}$$

The Verlet algorithm has some disadvantages. First, the velocity terms do not explicitly appear in the Verlet integration algorithm. This makes it difficult to obtain the velocities and they are only available once the positions have been computed at the next step. Second, it is not a self-starting algorithm. The velocities at time t are available only once the positions at time $t+\Delta t$ have been calculated. This means that at the beginning of the simulation, i.e. t=0, it is necessary to use a different formula or use the Taylor series of Eq. (1.112) truncated after the first term. Third, using the Verlet algorithm could result in loss of precision. This is so because the very small term, $\mathbf{a}(t)\Delta t^2$, is being added to the difference of two larger terms, namely, $2\mathbf{r}(t)$ and $\mathbf{r}(t-\Delta t)$. To avoid these problems, the Verlet algorithm was modified and used by Swope, Andersen, Berens and Wilson [96] in the computer simulation of physical clusters of molecules to include the velocity in the Verlet algorithm; this approach is known as the ***velocity Verlet algorithm***.

The velocity Verlet algorithm produces results that are equivalent to those of the standard Verlet algorithm. It gives position, velocities (needed for the calculation of temperature), and accelerations at the same time without compromising precision:

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 && (1.119)\\
\mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \frac{1}{2}[\mathbf{a}(t) + \mathbf{a}(t+\Delta t)]\Delta t. && (1.120)
\end{aligned}$$

There are variations of the Verlet integration algorithm that have been developed. Among them are the *leap-frog* [97] algorithm:

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t+\frac{1}{2}\Delta t)\Delta t && (1.121)\\
\mathbf{v}(t+\frac{1}{2}\Delta t) &= \mathbf{v}(t-\frac{1}{2}\Delta t) + \mathbf{a}(t)\Delta t && (1.122)
\end{aligned}$$

and the *Beeman's* [98] algorithm:

$$\begin{aligned}
\mathbf{r}(t+\Delta t) &= \mathbf{r}(t) + \mathbf{v}(t)\Delta t + \frac{2}{3}\mathbf{a}(t)\Delta t^2 - \frac{1}{6}\mathbf{a}(t-\Delta t)\Delta t^2 && (1.123)\\
\mathbf{v}(t+\Delta t) &= \mathbf{v}(t) + \frac{1}{3}\mathbf{a}(t)\Delta t + \frac{5}{6}\mathbf{a}(t)\Delta t - \frac{1}{6}\mathbf{a}(t-\Delta t)\Delta t && (1.124)
\end{aligned}$$

These different integration schemes are commonly used when performing molecular dynamics simulations.

1.8.2 Performing Molecular Dynamics Simulations

Figure 1.1 shows the general scheme for performing a typical molecular dynamics simulation. Several things must be considered and given attention in each step. Defining the composition of the system requires giving the number of atoms and their atomic masses. The atom types for each atom in the system must be explicitly declared including the interaction potential, whose definitions depend on the force-field employed.

The length of the simulation and the time step used must be defined. In choosing the appropriate time step, one has to consider the factors that limit the upper size of the time step. In general, the time step should be as large as possible so that the simulation is as long as possible, but not so large that accuracy of the integration algorithm is compromised. Usually, it is the nature of the highest frequency motions in the system, typically vibrational stretching mode involving carbon–hydrogen bonds, that is used as a guide for deciding

1. Define the composition of the system.
2. Choose the time step, Δt, and the length of the simulation.
3. Assign initial values to position $\mathbf{r}_i$, and velocities, $\mathbf{v}_i$, to atom i at $t = 0$.
4. Calculate the forces, $\mathbf{f}_i$, and acceleration, $\mathbf{a}_i$, at $t = 0$.
5. Calculate $\mathbf{r}_i$, $\mathbf{v}_i$, and $\mathbf{f}_i$ for the atom i at $t = t + \Delta t$.
6. Increment the time by Δt.
7. Repeat 5, until length of simulation is reached.
8. Analyze the results.

Figure 1.1: A general algorithm for molecular dynamics simulation.

the size of the time step. In practice, a time step of 1 fs is sufficient and often employed.

The initial configuration of the system, i.e. positions, may be taken from configurations determined by experimental data, from those determined by theoretical models, or from both. The initial values for the velocities can be determined from a Maxwell-Boltzmann velocity distribution:

$$f(v)dv = \left(\frac{m_i}{2\pi k_B T}\right)^{1/2} exp\left[-\frac{1}{2}\frac{m_i}{k_B T}v^2\right] dv \tag{1.125}$$

where $f(v)$ is the probability of each component of the velocity v between v and $v + dv$ for atom i of mass m_i at temperature T. The Maxwell-Boltzmann distribution is a Gaussian distribution which has a mean value of zero and a standard deviation of $\left(\frac{k_B T}{m_i}\right)^{1/2}$. This means that the values of the velocities of the atoms can be assigned by treating them as independent Gaussian random variables which in turn can be generated using a random number generator. Since the velocities are randomly generated, the temperature of the system will not be exactly T, but can be forced to equal T by using the *instantaneous temperature* of the system at time t:

$$T(t) = \frac{2}{(3N - N_c)k_B}K \tag{1.126}$$

where $(3N - N_c)$ is the total number of degrees of freedom accessible to the system, N_c is the number of constraints, and K is the instantaneous kinetic energy related to particles' momenta:

$$K = \sum_{i=1}^{N} \frac{|\mathbf{p}_i|^2}{2m_i}. \tag{1.127}$$

Eq. (1.126) is taken directly from the result of statistical thermodynamics that the temperature of the system is related to the time average of the kinetic energy:

$$T_{sys} = \frac{2}{(3N - N_c)k_B} \langle K \rangle . \tag{1.128}$$

Eq. (1.128) is an ensemble average that is done over all the configurations that are accessible to the system.

During the course of the simulation, controlling the temperature and/or pressure of the system is also important. For example, the temperature can be controlled by scaling the velocities of the particles or by coupling the system to some external bath. The pressure can be controlled by modification to the equation of motion for the coordinates and volume of the system. These topics are discussed next.

1.8.3 NVE, NVT and NPT Molecular Dynamics

Molecular dynamic simulation in which no temperature modification is done is a kind of simulation that is said to have been performed in the constant *NVE ensemble.* Simulations done at constant NVE require that the number of atoms or particles (N), volume (V) and energy (E) remain constant throughout the simulation. The other two common ensembles are the constant NVT and the constant NPT ensembles, where T and P refer to the temperature and the pressure, respectively.

In the constant NVT ensemble, one way to control the temperature is to do a simple scaling of the velocities by multiplying the velocities at each time step by a scaling factor, $\lambda = \sqrt{T_{sys}/T(t)}$, where T_{sys} is the desired system temperature and $T(t)$ is the current temperature at time t calculated from Eq. (1.126). An alternative way to control the temperature is to use the one proposed by Berendsen and co-workers [99].

In the Berendsen method, the system is coupled to an external heat bath that is fixed at the desired temperature, T_{bath}. The external heat bath, whenever appropriate, supplies to or removes from the system thermal energy. In this method, the rate of change of the temperature is proportional to the difference in the bath and system temperatures and the system is exponentially decaying towards the desired temperature:

$$\frac{dT(t)}{dt} = \frac{1}{\tau}(T_{bath} - T(t)). \tag{1.129}$$

The τ in Eq. (1.129) is the coupling parameter with units of time. The value of τ determines how tightly the bath and the system are coupled together. Large values of τ mean the coupling is weak, whereas small values mean the coupling is strong. At each time step during the simulation, the velocities are scaled by a scaling factor in the form of:

$$\lambda = \sqrt{1 + \frac{\Delta t}{\tau}\left(\frac{T_{bath}}{T(t)} - 1\right)}, \tag{1.130}$$

and the change in temperature between successive time steps is calculated as:

$$\Delta T = \frac{\Delta t}{\tau}(T_{bath} - T(t)). \tag{1.131}$$

The equations that govern the constant NPT ensemble are similar to those of the constant NVT ensemble. The major difference between the two is that the control of pressure in the NPT ensemble is associated with the fluctuation of the unit volume of the system. The amount of volume fluctuation is normally related to the *isothermal compressibility factor*, κ:

$$\kappa = -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_T. \tag{1.132}$$

In molecular dynamics, κ can be expressed in terms of the mean square volume displacement:

$$\kappa = \frac{1}{k_B T}\frac{\langle V^2\rangle - \langle V\rangle^2}{\langle V^2\rangle} \tag{1.133}$$

The larger the value of κ, the more compressible the substance is, hence larger volume fluctuations can occur at a given pressure.

Analogous to the Berendsen method for the temperature, the rate of change of pressure can be expressed as:

$$\frac{dP(t)}{dt} = \frac{1}{\tau_P}(P_{bath} - P(t)) \tag{1.134}$$

where τ_P is the pressure coupling constant in units of time, P_{bath} is the reference or 'bath' pressure, and $P(t)$ is the instantaneous or the actual pressure at time t. The scaling factor λ, which is related to the isothermal compressibility factor, κ:

$$\lambda = 1 - \kappa\frac{\Delta t}{\tau_P}(P(t) - P_{bath}) \tag{1.135}$$

is used for either adjusting the volume of the simulation box or for scaling the atomic coordinates:

$$\mathbf{r}_i^{'} = \lambda^{1/3}\mathbf{r}_i \tag{1.136}$$

where $\mathbf{r}_i^{'}$ are the new atomic positions, in order to maintain constant pressure.

1.9 Hybrid QM/MM–Molecular Dynamics

Conformational changes in biomolecular entities, such as proteins and enzymes, are often caused by biological processes that can alter the configuration at a specific site in the protein/enzyme. The specific alteration of the configuration may or may not lead to a major conformational change of the entire structure of the protein or enzyme. The conformational rearrangement can be caused by either the interaction of the amino acids that make up the protein/enzyme or by the interaction of small molecules with specific amino acids in the protein/enzyme. Typically, the interactions of these small molecules with the proteins/enzymes may be a non-covalent one or those that involve chemical reactions characterized by bond breaking and bond formation.

The discrete process involving chemical reactions is best described by the use of QM methods. In dealing with proteins/enzymes or any large biomolecular system for that matter, the use of QM techniques to represent the entire system is computationally very expensive, if not impossible. For this reason, QM techniques are applied to the small portion of the system of interest.

This portion may include ligand molecules, which may be involved in chemical/enzymatic reactions, plus the immediately surrounding part of the large biomolecular systems such as amino acids in proteins/enzymes. The rest of the system is treated by MM techniques. This is known as QM/MM technique described in section 1.5. However, biological processes are far from thermodynamic equilibrium and are very dynamic. In dealing with the dynamic nature of biological processes, a time coordinate must be explicitly included in the model. This can be addressed using molecular dynamics methods.

To simply state, in the hybrid QM/MM–MD approach the MD methods and techniques are utilized to follow the time evolution of the biomolecular system, which is treated with the QM/MM technique. The energy of the QM/MM system is calculated using Eq. 1.97 and the equations of motion are integrated using integration algorithms as discussed in section 1.8.1.

1.10 Solvents and the Free Energy of Solvation

Biochemical processes take place in the presence of solvent. To understand various phenomena concerning the stabilities of molecule-molecule association such as protein-protein and protein-ligand interaction, it is important to consider how the solvent affects the behaviour of a system.

There are several ways in which the solvent molecules may be represented in molecular simulations. If the solvent molecules are directly involved in chemical reactions or they are tightly bound to the solute, solvent molecules should be modeled explicitly. This approach consists of constructing detailed atomic models of the solvated system and literally following the dynamic motions of all the atoms in the system at some computational expense. For example, explicit water can be represented by TIPnP [86] and SPC [87] models. However, if solvent molecules do not interact directly with the solute but only provide an environment that strongly affects the behaviour of the solute, an implicit model of the solvent that mimics the behaviour of a bulk medium would be adequate. Some of the commonly used implicit models for the solvent include the *self-consistent reaction field* (SCRF) [100, 101], *polarizable continuum model* (PCM) [102], and the *conductor-like screening model*

(COSMO) [103, 104, 105].

It is not the intent of this thesis to discuss the models and methods for solvation. However, it is important to mention that solvent effects on the solute are generally measured quantitatively by the free energy of solvation, ΔG_{sol}. ΔG_{sol} is the free energy change required to transfer a molecule from the gas-phase to solution-phase. The solvation free energy consists of several components as given in Eq. (1.137):

$$\Delta G_{sol} = \Delta G_{elec} + \Delta G_{vdW} + \Delta G_{cav} + \Delta G_{hb} \tag{1.137}$$

where ΔG_{elec} is the electrostatic component representing the polarization of the solvent, ΔG_{vdW} is the van der Waals interaction between the solute and the solvent, ΔG_{cav} is the free energy associated with the formation of the solute cavity within the solvent, and ΔG_{hb} is the explicit hydrogen-bonding contribution term if there is a localised hydrogen bonding between the solute and the solvent.

Chapter 2

Genetic Algorithm: Theory and Programming

The concept of biological genetics, natural selection, evolution and adaptation as an inspiration for solving many computational problems is widespread. The idea of implementing such concept into computer programs falls under the category of probabilistic optimization algorithms, known as *evolutionary algorithm* (EA), an algorithm based on the model of biological evolution. There are three basic classes of evolutionary algorithm: evolution strategy (ES), evolutionary programming (EP) and genetic algorithm (GA). Among the three classes, GA is the most popular and the topic that will be discussed here.

Genetic algorithms have been used in computational modeling, especially in computational problems that would require searching for a huge number of possibilities for their solutions. The GA has been used and applied in a variety of problems: searching for the global minimum energy conformation of a molecule [106, 107, 108], conformational search of macromolecules like DNA [109] and RNA [110, 111], protein-folding [112, 113, 114, 115, 116, 117, 118, 119], molecular docking and molecular design [120, 121, 122]. The use of the genetic algorithm as a computational tool is an idea that stems from the concept that in nature every living organism has a unique set of chromosomes. Biologically, a chromosome is a long molecule of DNA, a hereditary material in humans and almost all other organisms. Different parts of a chromosome describe how to build a specific component of an organism. The different sections of the chromosomes are called genes. During sexual reproduction, the chromosomes from two parents are combined to produce offspring. Each offspring would have a mixture of the parents' chromosomes. The offspring may carry a combination of good or bad genes from the crossover of the chromosomes of their parents. The combinations of good genes could result in the production of offspring that are in some sense better or worse than their parents. Naturally, organisms that produce offspring carrying better genetic information than others tend to be fitter and will generally survive in nature. However, those offspring carrying inferior genetic material may still be able to survive or may be even better when compared to those carrying the more superior genetic informations, due to variations in the genes caused by mutation. In the course of natural selection, the individuals carrying better adapted genes tend to survive and spread through the population. The result would be a population consisting of more organisms that are well adapted to their environment.

The original genetic algorithm (GA) was proposed and developed by Holland [123] with his students and colleagues at the University of Michigan in the 1970s. Holland's idea was to develop ways in which the mechanism of natural adaptation could be implemented into computer systems, using processes such as selection, recombination, mutation and evaluation. The principles, variations and extensions are also described in books by Goldberg [124], Davis [125], and Michalewicz [126]. Although there have been variations from the original GA by Holland, the idea of GA remains the same and is in parallel with the biological evolution. The genetic algorithm finds potential solutions to the problem in which these solutions are found in the population, a population consisting of organisms or individuals. In short, each of the individuals carry a potential solution to the problem. The genetic algorithm can find better solutions which are equivalent to fitter organisms or individuals in nature. The genetic algorithm uses operations that allow it to pass genetic information from the present to the next generation.

In this thesis, only the basic genetic algorithm is discussed together with its implementation into computer program for use in the optimization of semi-empirical parameters for carbohydrate molecules. Since in GA an individual is defined by a chromosome, the use of the words *individual* and *chromosome* are interchangeable in the following discussions.

2.1 The Genetic Algorithm

There are fundamental issues that need to be addressed when using a genetic algorithm: representation of the chromosome, creation of the initial population, the evaluation function that determines the fitness of the chromosome, selection criteria, genetic operators such as reproduction and mutation that alter the composition of the population, and termination criteria. Conceptually, these issues can be represented by the following algorithm:

```
Begin at time $t = 0$
Generate initial population $P$ at $P(t = 0)$
Evaluate $P(t = 0)$
while $(P(t) \neq$ converged) do
    $t = t + 1$
    Select $P(t + 1)$
    Reproduce
    Mutate
    Evaluate $P(t + 1)$
end do
End
```

Figure 2.1: An outline of algorithm for a GA program.

2.1.1 Chromosome Representation

In all genetic algorithms, a representation for the chromosome is required to describe each individual in the population. The chromosome representation can be made up of sequence of genes, using letters of alphabet or numbers. The chromosome representation can consist of any of the following: binary digits (0 and 1), strings of binary digits, integers, floating point numbers, characters of the alphabet, matrices, and so on. In the original design by Holland, chromosome representation was limited to binary digits (0 and 1). Figures 2.2 and 2.3 shows examples of chromosome representation, where each value can represent the parameters to be optimized in GA.

Chromosome	1	1	1	1	0	0	0	1	1	0

Figure 2.2: A binary encoding of the chromosome.

Chromosome	1.0	3.5	6.8	-0.8	-1.3	-4.2	8.0	9.5

Figure 2.3: A real-valued encoding of the chromosome.

2.1.2 Creation of the Initial Population

Before a genetic algorithm can be used for evolving the solutions to better ones, an initial population of solutions must be present. The most common technique in generating an initial population is to just randomly generate solutions for the entire population. Other techniques that can be used are using specifically designed algorithms to produce the solutions or the use of previously evolved solutions. A combination of the techniques mentioned can also be used. In creating an initial population, the idea is to maximize the diversity of the initial solutions in the population and to introduce right away potentially good solution to population in order to have a higher chance of finding better solutions to the problem. It must be noted that in GA terms, a solution is represented by an individual or a chromosome, which in turn is used for function evaluation.

2.1.3 The Fitness Function

The fitness or evaluation function is used to evaluate a chromosome. It takes the genes making up the chromosomes to form a single solution. The fitness function usually returns a number which indicates how good the solution is. The individual values returned by the fitness function are meaningless unless used for comparing to other values returned by all the possible solutions. It is only then that better solutions can be selected. It must be noted that the fitness function is independent of the genetic algorithm.

2.1.4 Selection Operation

The selection of individuals that will serve as parents during the reproduction stage of the GA can be accomplished by using several schemes. These schemes, to name a few, include probabilistic selection methods like the roulette wheel selection and ranking methods, and other models such as the tournament and the elitist models.

In the probabilistic selection method, the fitness of an individual is used in the selection. All individuals in the population have a chance of being selected to crossover or reproduce to generate the next generation. However, the better individuals have higher chances of being picked and an individual can be selected more than once. An example of selecting an individual using a simple probabilistic selection is to generate a series of random numbers and then compare those numbers against a cumulative probability of the population.

In the roulette wheel selection, each solution in the population can be visualized as a slice of a roulette wheel. The probability, P_i, that an individual is chosen using the roulette wheel method can be defined as:

$$P_i = \frac{F_i}{\sum_{i=1}^{PS} F_i} \tag{2.1}$$

where F_i is the fitness of individual i, and PS is the population size. In this method, better solutions will be assigned proportionately larger sections of the wheel and therefore these solutions would have a higher chance of being selected. The selection can be visualized in the following manner. If two solutions are required, the wheel is spun twice. The selected solution can be removed from the wheel before the next selection or left on the wheel. Leaving good solutions on the wheel increases their chances of being picked again. This means that, just like in the simple probabilistic model, an individual can be selected more than once.

In the ranking method, the probability, P_i, is assigned based on the rank of solution i when all of the solutions are sorted. An example is normalized geometric ranking proposed by Joines and Houck [127] in which the probability, P_i, for each individual can be defined as:

$$P_i = \frac{q}{1-(1-q)^{PS}}(1-q)^{r-1} \tag{2.2}$$

where q is the probability of selecting the best individual, r is the rank of the individual (1 being the best), and PS is the population size.

The tournament selection method works like a real tournament among athletes. The selection is done by first selecting individuals randomly; then tournament strategy is used, where a number of rounds are played. Replacement of weak individuals and insertion of the best ones are allowed into the new population.

Elitist model is a strategy that guarantees the survival of the best individual for the next generation. Elitism can rapidly increase the performance of GA since it prevents losing the best solution throughout the evolution. It can be said that elite individuals or chromosomes are the best solutions for the current generation that are carried over in the next generation of a GA cycle.

2.1.5 Reproduction or Crossover Operation

Once the good parents have been selected, recombination techniques for crossing over the parents in a manner that will produce good and new offspring are needed. These techniques in GA are often referred to as crossover or reproduction operations. There are several ways in which crossover can be performed. Among these are *single-point crossover*, *two-point crossover*, and *uniform crossover*.

In the single-point crossover method, a crossover point is chosen randomly. The offspring are reproduced by exchanging the genes of the parents before and after the crossover point. For example, if the randomly chosen crossover point is 4, then the chromosome for the Offspring 1 is constructed by copying part of the chromosome of Parent 1 before the crossover point and then copying part of the chromosome of Parent 2 to and after the crossover point. The reverse is done for the chromosome of Offspring 2. This is illustrated in Figure 2.4.

The two-point crossover is very similar to the single-point crossover except that two crossover points are chosen randomly. The result is that Offspring 1 carries the head and the tail of the chromosome of Parent 1 while gaining the mid-section of the chromosome of the Parent 2. The Offspring 2, on the other hand, carries the head and tail of the chromosome of Parent 2 and the mid-section of the chromosome of the Parent 1. This is illustrated in Figure 2.5 where the randomly chosen crossover points are 3 and 8:

Parent Chromosomes

Parent 1	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Parent 2	0	1	2	3	4	5	6	7	8	9

Offspring Chromosomes

Offspring 1	**A**	**B**	**C**	**D**	4	5	6	7	8	9
Offspring 2	0	1	2	3	**E**	**F**	**G**	**H**	**I**	**J**

Figure 2.4: A single-point crossover.

Parent Chromosomes

Parent 1	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Parent 2	0	1	2	3	4	5	6	7	8	9

Offspring Chromosomes

Offspring 1	**A**	**B**	**C**	3	4	5	6	7	**I**	**J**
Offspring 2	0	1	2	**D**	**E**	**F**	**G**	**H**	8	9

Figure 2.5: A two-point crossover.

In the uniform crossover, a certain n number of points in the chromosomes are selected, where n is a random number which is less than the chromosome length. Each selected point is swapped over to create the offspring. An illustration is given in Figure 2.6

Parent Chromosomes

Parent 1	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Parent 2	0	1	2	3	4	5	6	7	8	9

Offspring Chromosomes

Offspring 1	0	**B**	**C**	3	4	5	**G**	**H**	**I**	**J**
Offspring 2	**A**	1	2	**D**	**E**	**F**	6	7	8	9

Figure 2.6: A uniform crossover.

2.1.6 Mutation Operation

In nature, mutation helps maintain the diversity of the population. Mutation or error in the genes can occur in individuals. The mutations cannot be produced by reproduction or crossover alone. These mutations can sometimes result in good features of the individual. These good features may be carried in the future generations and can eventually lead to the creation of better individuals.

In genetic algorithm, the mutation in nature is imitated by forcing errors in the chromosomes. This is accomplished by changing one or more points in the chromosome of the parents randomly by some valid values. The mutation operation in GA is dependent on how the chromosomes are encoded or represented. Examples are given in Figures 2.7 and 2.8. In both Figures, the mutation occurs at point 3 of the chromosome.

2.1.7 Termination of GA

Refering to Figure 2.1, the GA will continue to cycle moving from one generation to the next performing the operations selection, reproduction, mutation and fitness evaluations until termination criteria are met. One of the most

Original Chromosome	1	1	**1**	1	0	0	0	1	1	0
Mutated Chromosome	1	1	**0**	1	0	0	0	1	1	0

Figure 2.7: A mutation in the binary encoding of the chromosome.

Original Chromosome	1.0	3.5	**6.8**	-0.8	-1.3	-4.2	8.0	9.5
Mutated Chromosome	1.0	3.5	**7.5**	-0.8	-1.3	-4.2	8.0	9.5

Figure 2.8: A mutation in the real-valued encoding of the chromosome.

commonly used termination criterion is the maximum number of generations specified. Other termination methods make use of population convergence. Population convergence criteria usually make use of deviations of the fitness values of the individuals in the population. For example, when the standard deviation of the fitness values is smaller than a specified threshold, the GA can be terminated.

2.2 Program Description and Coding

Figure 2.9 illustrates the main program organization for the genetic algorithm used in this work. This program was written in Java [128] programming language for the present research project. The program organization follows the algorithm given in Figure 2.1. As previously mentioned, the genetic algorithm program is written for the optimization of PM3 parameters for the atoms present in carbohydrate molecules: hydrogen, carbon, and oxygen. Chromosomes are represented by real numbers, where each number corresponds to a PM3 parameter. In the following discussions, whenever appropriate, the piece or pieces of the codes relevant will be shown in italics.

2.2.1 The Zeroth Generation

Each generation is represented by the variable g. The program begins by setting the variable g to zero (the zeroth generation). Then, the parameters needed for performing the GA are read from a starter file. Figure 2.10 shows an example of the structure of the starter file. This file includes the following information:

1. the initial size of the population for the zeroth generation ($Population_{ini}$);
2. the size of the population for the succeeding generations ($Population_{size}$);
3. number of variables that require optimization ($N_{variables}$);
4. number of elite individuals to keep in the population for the next generation (N_{elites});
5. maximum number of generations to run the GA ($N_{generations}$);
6. the fraction of mutation that is allowed in the individual gene ($F_{mutation}$), and;
7. the list of initial values of the parameters.

The number of elite individuals and the fraction of mutation will be discussed later.

```
Set g=0, the zero^th generation
Read GA parameters and N generations
Generate initial GA population
Calculate the fitness values
Rank members of the generation
while g < N generations do
    Increment g
    Select parents
    Crossover parents
    Mutate population
    Calculate the fitness values
    Rank members of the g^th generation
    If converged, exit while loop
end do
Print report
```

Figure 2.9: The pseudocode for the main GA program.

$Population_{ini}$ $Population_{size}$ $N_{variables}$ N_{elites} $N_{generations}$ $F_{mutation}$
List of Initial Parameters
200 100 36 2 20 0.01
-12.415235
-5.167165
0.988622
$\vdots$
up to the 36^{th} variable ← this corresponds to $N_{variables}$

Figure 2.10: An example of the structure of the starter file.

Based on the value given for the *$Population_{ini}$* and using each initial value of the parameters to be optimized, an initial population for the GA is randomly generated and is then saved in a file. For example, if *$Population_{ini}$* = 200, then the initially generated population would consist of 200 chromosomes where each chromosome represents a possible solution. In the present case, each chromosome consists of the PM3 parameters that are randomly generated within ±5% of the standard PM3 values. An example of an initial population is given in Figure 2.11. The values in each chromosome given in Figure 2.11 correspond to the thirty-six PM3 parameters – ten for hydrogen, thirteen for carbon and thirteen for oxygen. The labels of the parameters within the GA program are given in Figure 2.12, where the USS(1) being the first parame-

$Population_{ini}$ $Population_{size}$ $N_{variables}$ N_{elites} $N_{generations}$ $F_{mutation}$
List of Parameters
200 100 36 2 20 0.01
-12.415235 -5.167165 ⋯ 5.950512 1.598395 ← 36 parameters in a row
-12.279855 -5.156250 ⋯ 5.780400 1.659466 ← a chromosome
-11.972161 -5.199856 ⋯ 6.106597 1.593319
$\vdots$ $\vdots$ $\vdots$ $\vdots$
up to the 200^{th} chromosome ← this corresponds to $Population_{ini}$

Figure 2.11: An example of a generated initial population.

ter and FN3(8,2) being the $36th$ parameter. The definitions of these labels can be found in Table 1.2 where the labels FN1, FN2, and FN3 correspond to the Gaussian parameters a, b, and c, respectively. These symbols also correspond to the symbols used in the "mopac_parameters" module in the DYNAMO [40] computer program. The values of these symbols are modified in the "mopac_parameters" module and are eventually used for evaluating the fitness of each chromosome. In the GA program, a chromosome is represented by a two-dimensional array, `c[i][j]`, where `i` is the size of the population, $Population_{size}$ (or $Population_{ini}$ at the zeroth generation), and `j` is the number of parameters, $N_{variables}$, to optimize.

USS(1)	BETAS(1)	ZS(1)	ALP(1)			
FN1(1,1)	FN2(1,1)	FN3(1,1)	FN1(1,2)	FN2(1,2)	FN3(1,2)	
USS(6)	UPP(6)	BETAS(6)	BETAP(6)	ZS(6)	ZP(6)	ALP(6)
FN1(6,1)	FN2(6,1)	FN3(6,1)	FN1(6,2)	FN2(6,2)	FN3(6,2)	
USS(8)	UPP(8)	BETAS(8)	BETAP(8)	ZS(8)	ZP(8)	ALP(8)
FN1(8,1)	FN2(8,1)	FN3(8,1)	FN1(8,2)	FN2(8,2)	FN3(8,2)	

Figure 2.12: The labels corresponding to each of the PM3 parameters used in GA. All parameters are placed in a single line.

The evaluation of the fitness function is independent of the genetic algorithm. The fitness of each chromosome of the initial GA population is evaluated by first calling the external DYNAMO program, making modifications to the "mopac_parameters" module and then rebuilding the DYNAMO library. DYNAMO then optimizes the geometry of the molecule using the current set of semi-empirical parameters and returns the optimized geometry, which in turn can be used to evaluate the fitness function. The fitness function can be defined in many possible ways, as long as it makes use of the semi-empirical PM3 method and its parameters. In the case of the reoptimization of the PM3 parameters for carbohydrates, the fitness function is calculated by comparing the relative energies and the geometry of the reference training set of monosaccharide molecules against those calculated from the current set of PM3 parameters (i.e. the current chromosome). The fitness function F is defined as:

$$F = w_E \sum_{i=1}^{M} \left(\Delta E_i^{MP2} - \Delta E_i^{PM3} \right) + w_S \sum_{i=1}^{M} S_{rms}^{(MP2,PM3)},$$

where w_E and w_S are arbitrary weight factors, M is the number of conformers of the monosaccharides, ΔE are the relative energies of the monosaccharide conformers, and $S_{rms}^{(MP2,PM3)}$ is the root-mean-square deviation of the calculated PM3 geometry of the monosaccharide from that of the MP2 reference geometry.

Once all the chromosomes are evaluated, the members of the population are ranked according to their fitness values from lowest to highest using a simple sorting algorithm. The chromosome with the lowest fitness value (the best chromosome) is assigned rank 0. The sorted initial population is used for preparing the population for the next generation.

2.2.2 The Subsequent Generations

The genetic algorithm is allowed to run for a specified N generations or until a certain convergence criterion is met. For the current generation g, the preparation of the population and evaluation of the individual members of the population make use of simple genetic algorithm operations, namely the *selection*, *crossover*, and *mutation*. The size of the population for each cycle of the genetic algorithm is given by $Population_{size}$ in Figure 2.11.

Selection operation. The selection of the chromosomes used in this work is very simple. The best half of the population of the previous generation is selected, i.e., the individuals belonging to one-half of the population with the best fitness values are selected. In the program, this is done by copying the chromosomes of this first one-half of the population into the current generation. With this criterion for selection, the best chromosome (i.e. ranked 0) is always selected. Additionally, the duplication of chromosomes is possible, meaning that the same individual may be included more than once in the current population. This is allowed since the chromosomes will be subjected to mutations later on that can alter their fitness values.

Crossover operation. So far, only half of the entire population is generated, via selection, for the current generation. The other half needs to be taken or generated somewhere else. The creation of the other half of the population is accomplished through a crossover or a reproduction procedure. The method used in this work is a single-point crossover of n-parents to produce n-offspring. For example, two parents, after performing a single-point crossover, will produce two offspring as illustrated in Figure 2.4. So, with the first best half of the population chosen in the selection procedure above, a single-point crossover procedure will produce the other half of the population for this generation. Figure 2.13 shows the segment of the code for performing single-point crossover. The crossover point, `ncut`, where the chromosome is cut, is determined randomly.

```
// Method to perform single-point crossover of the first best
// Population_size/2 parents to produce Population_size/2 kids.
   .
   .
   .
   Random random = new Random();
   int k = 0;
   for (int i=Nkids; i<Nkids+Nkids/2; i++) {
       // ncut is the crossover point
       int ncut = 1 + (int)(Nvar * random.nextDouble());
       for (int m=0; m<ncut; m++){
           c[i][m] = c[k][m];
           c[i+Nkids/2][m] = c[k+1][m];
       }
       for (int m=ncut; m<Nvar; m++){
           c[i][m] = c[k+1][m];
           c[i+Nkids/2][m] = c[k][m];
       }
       k += 2;
   }
   // sort and rank;
   .
   .
   .
```

Figure 2.13: The code for the crossover of the GA program.

Once the offspring are generated, the population becomes equal to the $Population_{size}$. The new population with size, $Population_{size}$, is then sorted and the fitness values are ranked from lowest to highest. This new population can now be subjected to the mutation operation, which is described next.

Elite individuals and mutation operation. As mentioned before, the elite individuals carry the best solutions for the current generation in a GA cycle. The computer code in Figure 2.14 allows the genes of a certain number of elite individuals (defined as N_{elites} in Figure 2.10) to be mutated by a certain fraction, `Fmutation` (defined as $F_{mutation}$ in Figure 2.10). The number of genes in an elite chromosome that is allowed to be mutated is determined by the variable `mb`. For example, if there are 36 parameters ($N_{var} = 36$) and the fraction of mutation is 0.01, then `mb` or the number of genes to mutate in a chromosome is 1. The position of this particular gene in the chromosome is then determined randomly, and is given by the variable `ib`. Once the position of the gene in the chromosome is determined, mutation is performed by randomly selecting a replacement value, `r[ib]`. In this case, the replacement value is to within the ±10% of the original parameter value. The fitness of the new chromosome carrying the mutated gene is then evaluated. If the fitness of the new chromosome is better (i.e. lower fitness value) than the fitness of the chromosome having the unmutated gene, then the new chromosome is considered more elite and replaces the other elite chromosome it is derived from.

Mutations in the other individuals that are not considered elite are also performed. Recall in Section 2.2.1 that the chromosomes are represented by a two-dimensional array variable, `c[i][j]`. The mutation is accomplished by first selecting the indices `i` and `j`, randomly. Similar to the mutation of the elite individuals, the chosen gene is replaced with a value to within the ±10% range.

2.2.3 The GA Termination

The GA program can be terminated in two ways: (1) once the number of specified generations, $N_{generations}$, is reached or (2) through the evaluation of the standard deviation of the current generation. The first one is straightforward. The lower the value set for the $N_{generations}$, the faster the termination of

```
// Mutation in the elite individuals
    .
    .
    .
    Random random = new Random();
    double[] r    = new double[Nvar];
    double[][] d  = new double[Nelite][Nvar];

    for (int i=0; i<Nelite; i++) {
        for (int m=0; m<Nvar; m++) r[m] = c[i][m];

        // mb is the number of genes to mutate
        int mb = (int)(Nvar*Fmutation+1);

        for (int j=0; j<mb; j++){
            // ib is the position in the chromosome to mutate
            int ib = (int)(Nvar * random.nextDouble());

            // r[ib] is the replacement gene in the ib position
           r[ib] = ((random.nextDouble()/5.0 + 0.90)) * r[ib];
        }
        for (int m=0; m<Nvar; m++) d[i][m] = r[m];
            Dynamo.makeLibrary(i, Nvar, d);
            double fn = Dynamo.calcFitnessFunction();
            if (fn < f[i]){
                for (int m=0; m<Nvar; m++) c[i][m] = r[m];
                f[i] = fn;
            }
        }
    .
    .
    .
```

Figure 2.14: The code for the mutation of the elite individuals of the GA program.

```
// Mutation in the other individuals
   .
   .
   .
   Random random = new Random();
   double[] r    = new double[Nvar];
   double[][] d  = new double[Nelite][Nvar];

   int mmax = (int)((Poolsize-Nelite)*Nvar*Fmutation+1);
   for (int i=0; i<mmax; i++) {
      int ig = (int)( (Poolsize-Nelite)*random.nextDouble()
                      + Nelite);
      int ib = (int)(Nvar * random.nextDouble());
      c[ig][ib] = ((random.nextDouble()/5.0 + 0.90)) * c[ig][ib];
   }
   .
   .
   .
```

Figure 2.15: The code for the mutation of the other individuals of the GA program.

the GA, and vice versa. However, the termination condition using $N_{generations}$ may neither give nor guarantee that the best possible solution is found, especially if the value given to $N_{generations}$ is low. The combination of a high value for the $N_{generations}$ and second criterion is more flexible and may be a better choice. In this case, the program can either terminate as soon as the specified standard deviation is reached or by giving sufficient time for the GA to finish up to the specified $N_{generations}$.

The program segment for the second criterion is shown in Figure 2.16. The standard deviation for the current generation is calculated by taking the square root of the population variance, `PoolVariance`. The `PoolVariance` is calculated by using the sum of the square of the fitness values of the individuals in the population (`var`), the average fitness of the population (`PoolAve`), the sum of the fitness values of the population (`totalOfFitness`), and the population size (`Poolsize`).

2.3 Conclusion

One of the reasons for the success of the GA in parameter optimizations is that the principles governing the algorithm are simple enough so that it can be easily understood. There are many factors that can affect how the GA can search through a set of possible solutions. The operations for the creation of the initial populations, selection techniques, reproduction and mutation methods can all be designed from very simple ways to more complex and sophisticated ones.

In this work, the presented computer codes necessary for performing a GA run are very simple. Improvements to the existing computer codes presented and/or programming additional computers codes for the other GA operations described in Section 2.1 are suggested. Although the codes are simple, using them for the optimization of semi-empirical parameters, presented in the next chapter, is adequate.

```
// Method to calculate the standard deviation
   .
   .
   .
   double var = 0.0;

   // Accumulate variance
   for (int i=0; i<Poolsize; i++ ) var += f[i]*f[i];

   PoolAve = totalOfFitness()/Poolsize;

   // variance
   PoolVariance = (var-(PoolAve*totalOfFitness()))/(Poolsize-1);

   if (PoolVariance <= 0.0) {
      PoolVariance = 0.0;
      PoolStdDev   = 0.0;
   }
   else {
   // standard deviation
   PoolStdDev = Math.sqrt(PoolVariance);
   }
   .
   .
   .
   if (PoolStdDev <= threshhold) {
      System.out.println("GA is converged...");
   }
   .
   .
   .
```

Figure 2.16: The code for calculating the standard deviation in the GA program.

Chapter 3

Genetic Algorithm: Application to Semi-empirical PM3 parameters for Monosaccharides

Modeling large complex systems such as carbohydrates requires an accurate description of a range of intricate and highly specific inter- and intramolecular interactions. Many complex carbohydrate parameter sets for calculations have been developed, mainly force-fields for molecular mechanics (MM). These force-fields are incorporated in computer programs such as AMBER [129, 130, 131, 132], CHARMM [133, 134], and GROMOS [135]. Typically, the parameterization procedure for creating the MM force-fields uses experimental X-ray structures of carbohydrate molecules, or *ab initio* calculations of fragment molecules that are considered building blocks for carbohydrate structures. However, no matter how complex these force-fields are, these are fixed-charge models. In order to model the intricate nature of the carbohydrates, one has to consider in the model the incorporation of the dynamic distribution of electrons in the system under investigation. This can be accomplished by directly performing simulations using quantum mechanical (QM) or density functional theory (DFT) techniques. However, due to the size and complexity of carbohydrates, high level QM methods are not feasible and DFT may be limited to treating molecules containing about 50 atoms. An alternative is to use methods that incorporate much faster computation than traditional QM methods but at the same time consider relatively accurate description of electronic structure in the system. One such method is the semi-empirical approach.

Semi-empirical methods, such MNDO [44], AM1 [50], and PM3 [52], have been originally designed to model small organic molecules. These methods are now commonly used in the study of molecular systems containing mostly the main-group elements [53, 136, 137], where the ground-state valence electronic configuration can be described by s and p or even d atomic orbitals [46, 47, 138, 139, 49]. Within the AM1 and PM3 formalism, the core-core repulsion interactions are expressed as modification to the core-core repulsion function of the MNDO method:

$$\begin{aligned} E_{AB}^{AM1,PM3} &= E_{AB}^{MNDO} + \frac{Z_A Z_B}{R_{AB}} \\ &\times \left(\sum_i a_{iA} e^{-b_{iA}(R_{AB}-c_{iA})^2} + \sum_j a_{jB} e^{-b_{jB}(R_{AB}-c_{jB})^2} \right) \end{aligned} \tag{3.1}$$

where

$$E_{AB}^{MNDO} = Z_A Z_B (s_A s_A | s_B s_B)(1 + e^{-\alpha_A R_{AB}} + e^{-\alpha_B R_{AB}}). \tag{3.2}$$

The Z_A and Z_B correspond to the core charges and R_{AB} is the internuclear separation. The parameters a, b, and c in Eq. (3.1) describe Gaussian functions centered at various distances c that modify the interaction between the two atoms. The α_A and α_B in Eq. (3.2) are some adjustable parameters. If $-OH$ and $-NH$ bonds are present, the modified E^{MNDO} given in Eq. (3.3) is used in Eq. (3.1) instead.

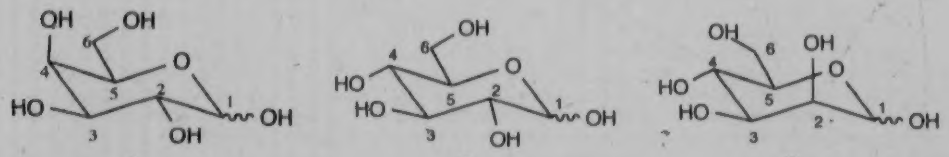
$$E_{XH}^{MNDO} = Z_X Z_H (s_X s_X | s_H s_H)(1 + R_{XH} e^{-\alpha_X R_{XH}/R_{AB}} + e^{-\alpha_H R_{XH}}). \tag{3.3}$$

In this chapter, the new semi-empirical PM3 parameters are presented. These parameters were obtained by using monosaccharide molecules as a reference set instead of those obtained from parameterization using fragment molecules that are building blocks of monosaccharides. The conformers of the monosaccharide molecules were calculated using *ab initio* method.

3.1 Computational Methods

Pyranose rings in solution are relatively rigid and exist mainly in the ${}^4C_1$ chair conformation [140]. For this study, selected monosaccharides in the ${}^4C_1$ chair conformation were used as reference molecules for obtaining PM3 parameters for specific use in carbohydrate simulations.

In order to reparameterize selected PM3 parameters, a locally developed genetic algorithm (GA) that was interfaced to the DYNAMO [40] computer program has been used. The idea follows the specific reaction parameters (SRP) approach proposed by Rossi and Truhlar [141], where selected semi-empirical parameters were adjusted to fit high level quantum mechanical calculations data for a specific reaction. In this work, the selected semi-empirical PM3 parameters were adjusted to fit the energies and geometries of selected hexopyranose conformers that were calculated using a high level *ab initio* method. Figure 3.1 shows the structures of the hexopyranoses considered in this work. These consisted of 8 conformers of D-galactopyranose **1**, 8 conformers of D-glucopyranose **2**, and 6 conformers of D-mannopyranose **3**. The notation used for each of the ${}^4C_1$ conformation of the monosaccahrides was adapted from Kony *et al.* [142]. The label for each conformer indicates the following: anomer (α or β)/the orientation of the hydrogen-bond network (*cc*=counter-clockwise or *cl*=clockwise)/the O-$C5$-$C6$-O and the $C5$-$C6$-O-H dihedral angles (t=trans, g^+=gauche$^+$, or g^-=gauche$^-$). Prior to GA optimization of parameters, the geometry for each of these 22 hexopyranose conformers were optimized. The geometry optimizations were carried out with the GAMESS-US program [37] at the MP2 level [12] using the cc-pVTZ basis set [143]. The relative energies and optimized geometries of these monosaccharides were used as reference for obtaining the GA optimized PM3 parameter set.



1 2 3

Figure 3.1: The structures of the D-hexopyranoses.

During the genetic algorithm fitting procedure, the following PM3 parameters were optimized: [U_{ss}, β_s, ζ_s, α, a_1, b_1, c_1, a_2, b_2, c_2] for hydrogen, and [U_{ss}, U_{pp}, β_s, β_p, ζ_s, ζ_p, α, a_1, b_1, c_1, a_2, b_2, c_2] for carbon and for oxygen. In the PM3 method, the parameter U represents the atomic orbital one-electron one-center integral term, β is the atomic orbital one-electron two-center resonance integral term, ζ is the Slater-type atomic orbital exponent and α is the atom core-core repulsion term. The parameters a, b, and c have already been described above. For C, H, and O atoms, a total of 36 parameters are allowed to vary during the GA optimization. In order to start the genetic algorithm optimization, the standard PM3 parameters were used to generate the initial population allowing only up to ±5% variation from these standard values. The initial population typically varies and consists of between 100 to 150 chromosomes. The GA optimizations were performed between 20–40 generations using the error function F defined as

$$F = w_E \sum_{i=1}^{M-3} \left(\Delta E_i^{MP2} - \Delta E_i^{PM3} \right) + w_S \sum_{i=1}^{M} S_{rms}^{(MP2,PM3)},$$

where w_E and w_S are arbitrary weight factors, M is the number of conformers of D-galactopyranose, D-glucopyranose, and D-mannopyranose, ΔE are the relative energies of the monosaccharide conformers, and $S_{rms}^{(MP2,PM3)}$ is the root-mean-square deviation of the calculated PM3 structure of the monosaccharide from that of MP2. The weight factors w_E (in mol/kJ) and w_S (in Å^{-1}) were chosen to be both 100. The $S_{rms}^{(MP2,PM3)}$ was calculated after superimposing the reference MP2 and the calculated PM3 structures using Kabsch's method[144].

MP2 geometry optimizations using cc-pVTZ basis set were also performed for four pentapyranoses, namely, D-arabinopyranose **4**, D-lyxopyranose **5**, D-ribopyranose **6**, and D-xylopyranose **7**. Their structures are shown on Figure 3.2. Only the α and the β forms with the hydrogen bond network in the counterclockwise direction were considered. The relative energies and the overall structures of the pentapyranoses were used to gauge the performance of the GA-optimized PM3 parameters for monosaccharides outside the training set.

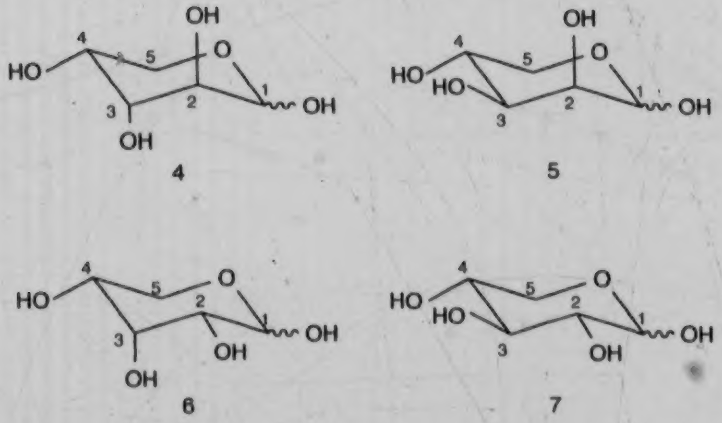


Figure 3.2: The structures of the D-pentapyranoses.

3.2 Results and Discussion

Tables 3.1–3.3 compare the GA-optimized PM3 parameters (denoted as $\mathrm{PM3}^{MS}$, MS for monosaccharide) and the standard PM3 ($\mathrm{PM3}^{STD}$). It may be seen that there are differences in the values of the parameters between $\mathrm{PM3}^{STD}$ and $\mathrm{PM3}^{MS}$. However, not all the PM3 parameters changed from the initial $\mathrm{PM3}^{STD}$ values. This is especially true for the parameters a, b, and c for oxygen. This simply means that $\mathrm{PM3}^{STD}$ a, b, and c parameters for oxygen are more or less already optimized in describing the monosaccharides used in the reference set.

As mentioned above, three hexopyranoses were used as the reference set each with varying number of conformations. Table 3.4 shows the relative energies of the different conformations of the three monosaccharides used in the training set. The lowest energy conformer for each monosaccharide is used as the reference point. The energies of the conformers for each of the monosaccharides are also arranged from lowest to highest to clearly see how close the energy gap among the conformers are.

$\mathrm{PM3}^{STD}$ parameterization correctly predicted the lowest energy conformer only for the D-galactose, the $\alpha/cl/g^{-}g^{+}$ conformer. None of the highest energy conformers were correctly predicted. The calculated r.m.s. deviations is the highest in D-glucose at 9.1 kJ/mol, with an overall r.m.s. deviation of 7.8

Table 3.1: $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for hydrogen.

Parameter	Unit	$PM3^{STD}$	$PM3^{MS}$
U_{ss}	eV	-13.073321	-12.436734
β_s	eV	-5.626512	-4.289542
ζ_s	bohr^{-1}	0.967807	0.988784
α	Å^{-1}	3.356386	3.118553
a_1	–	1.128750	1.127411
b_1	Å^{-1}	5.096282	5.094134
c_1	Å	5.096282	1.537465
a_2	–	-1.060329	-1.060329
b_2	Å^{-1}	6.003788	6.006872
c_2	Å	1.570189	1.570189

Table 3.2: $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for carbon.

Parameter	Unit	$PM3^{STD}$	$PM3^{MS}$
U_{ss}	eV	-47.270320	-44.415413
U_{pp}	eV	-36.266918	-37.396380
β_s	eV	-11.910015	-12.110500
β_p	eV	-9.802755	-9.549442
ζ_s	bohr^{-1}	1.565085	1.546361
ζ_p	bohr^{-1}	1.842345	1.778750
α	Å^{-1}	2.707807	2.814684
a_1	–	0.050107	0.048141
b_1	Å^{-1}	6.003165	4.142953
c_1	Å	1.642214	1.581081
a_2	–	0.050733	0.050727
b_2	Å^{-1}	6.002979	5.281214
c_2	Å	0.892488	0.890562

Table 3.3: $PM3^{STD}$ and optimized $PM3^{MS}$ parameters for oxygen.

Parameter	Unit	$PM3^{STD}$	$PM3^{MS}$
U_{ss}	eV	-86.993002	-82.154826
U_{pp}	eV	-71.879580	-74.660258
β_s	eV	-45.202651	-41.899362
β_p	eV	-24.752515	-28.450886
ζ_s	bohr^{-1}	3.796544	3.607678
ζ_p	bohr^{-1}	2.389402	2.221131
α	Å^{-1}	3.217102	3.071855
a_1	–	-1.131128	-1.131128
b_1	Å^{-1}	6.002477	6.002477
c_1	Å	1.607311	1.607311
a_2	–	1.137891	1.137891
b_2	Å^{-1}	5.950512	5.950512
c_2	Å	1.598395	1.598395

kJ/mol or 1.9 kcal/mol for the entire training set.

On the other hand, PM3^{MS} predicted correctly both the lowest and the highest energy conformers for D-galactose and D-glucose. The lowest energy conformer was also correctly predicted in D-mannose, but not the highest energy one. Even though the highest energy conformer in D-mannose was not correctly predicted, by looking at the energy differences in D-mannose between the ones predicted by MP2 and the PM3^{MS}, it is noticeable that the difference is very small. Based on the calculated average r.m.s. deviations for galactose, glucose, and mannose, respectively, all have essentially the same magnitude of error compared to the MP2 energy reference. The calculated overall r.m.s. deviation in the energy for PM3^{MS} is just merely 1.4 kJ/mol (0.3 kcal/mol).

Table 3.5 shows the error in the overall geometry for both PM3^{STD} and PM3^{MS} with respect to the MP2 geometry. As before, the average r.m.s. deviations are calculated for each monosaccharide to illustrate which of the monosaccharides is best represented structurally by both PM3^{STD} and PM3^{MS}. Although both sets of parameters are off by more than 0.10 Å, the PM3^{MS} shows a slight improvement over PM3^{STD}.

The recent PM6 [43] parameterization in MOPAC2007 [41] was also tested using the conformations of the three monosaccharides used in the training set

Table 3.4: Relative energies (in kJ/mol) of the different ${}^4C_1$ conformations of D-galactose, D-glucose and D-mannose and the r.m.s. deviations with respect to MP2/cc-pVTZ.

Conformer	MP2	PM6	PM3^{STD}	PM3^{MS}
D-galactose				
$\alpha/cl/g^-g^+$	0.0	0.0	0.0	0.0
$\alpha/cc/g^+g^-$	7.0	-0.5	8.1	6.7
$\alpha/cc/tg^+$	7.6	-0.1	11.8	8.0
$\alpha/cc/g^-g^-$	9.1	12.1	10.8	8.6
$\beta/cl/g^-g^+$	12.6	11.6	2.9	16.0
$\beta/cc/g^+g^-$	15.4	16.5	5.1	15.4
$\beta/cc/tg^+$	16.5	17.5	8.9	17.3
$\beta/cc/g^-g^-$	17.6	29.2	7.4	17.5
Average r.m.s.d.		5.7	6.9	1.3
D-glucose				
$\alpha/cc/g^-g^+$	0.0	0.0	0.0	0.0
$\alpha/cc/tg^+$	1.3	-4.0	-1.4	1.3
$\alpha/cc/g^+g^-$	1.4	-1.0	-1.9	1.2
$\alpha/cl/g^-g^+$	3.3	11.3	0.2	1.3
$\beta/cc/g^-g^+$	6.0	15.7	3.6	9.6
$\beta/cc/g^+g^-$	10.0	14.2	-3.3	10.0
$\beta/cc/tg^+$	11.5	12.6	-4.3	11.3
$\beta/cl/g^-g^+$	15.4	15.8	1.2	15.1
Average r.m.s.d.		5.1	9.1	1.4
D-mannose				
$\alpha/cl/g^-g^+$	0.0	0.0	0.0	0.0
$\beta/cl/g^-g^+$	1.6	16.6	-4.4	1.5
$\alpha/cl/tt$	4.7	5.1	7.7	7.4
$\beta/cl/tt$	5.4	20.6	2.7	8.0
$\beta/cl/g^+g^-$	7.8	13.4	-5.3	8.5
$\alpha/cl/g^+g^-$	8.1	-2.8	-0.1	8.1
Average r.m.s.d.		10.1	6.9	1.5
Overall r.m.s.d.		7.0	7.8	1.4

Table 3.5: S_{rms} deviations (in Å) for the $^4C_1$ conformations of D-galactose, D-glucose and D-mannose with respect to MP2/cc-pVTZ geometry.

Conformer	PM6	PM3^{STD}	PM3^{MS}
D-galactose			
$\alpha/cl/g^-g^+$	0.18	0.10	0.09
$\alpha/cc/g^+g^-$	0.21	0.15	0.11
$\alpha/cc/tg^+$	0.13	0.19	0.12
$\alpha/cc/g^-g^-$	0.11	0.09	0.09
$\beta/cl/g^-g^+$	0.26	0.14	0.12
$\beta/cc/g^+g^-$	0.24	0.19	0.12
$\beta/cc/tg^+$	0.18	0.23	0.14
$\beta/cc/g^-g^-$	0.20	0.17	0.15
Average r.m.s.d.	0.19	0.16	0.12
D-glucose			
$\alpha/cc/g^-g^+$	0.15	0.08	0.08
$\alpha/cc/tg^+$	0.14	0.08	0.11
$\alpha/cc/g^+g^-$	0.15	0.10	0.10
$\alpha/cl/g^-g^+$	0.24	0.08	0.08
$\beta/cc/g^-g^+$	0.23	0.10	0.11
$\beta/cc/g^+g^-$	0.21	0.11	0.10
$\beta/cc/tg^+$	0.19	0.14	0.12
$\beta/cl/g^-g^+$	0.96	0.11	0.11
Average r.m.s.d.	0.28	0.10	0.10
D-mannose			
$\alpha/cl/g^-g^+$	0.21	0.09	0.10
$\beta/cl/g^-g^+$	0.15	0.11	0.11
$\alpha/cl/tt$	0.17	0.09	0.11
$\beta/cl/tt$	0.12	0.09	0.11
$\beta/cl/g^+g^-$	0.67	0.17	0.12
$\alpha/cl/g^+g^-$	0.68	0.17	0.11
Average r.m.s.d.	0.33	0.12	0.11
Overall r.m.s.d.	0.26	0.13	0.11

to determine how well PM6 improved over PM3 for modeling these monosaccharides. PM6 showed very similar results as those obtained from using PM3. Neither the lowest nor the highest energy conformers were correctly predicted. In some cases the optimized geometries calculated were even worse than those of PM3 geometries as compared to the MP2 reference.

Tables 3.6 and 3.7 illustrate how the PM3^{STD} and PM3^{MS} compare with the MP2 reference for the D-pentapyranoses. Neither the lowest nor the highest energy conformers was predicted by the PM3^{STD}. The PM3^{MS} was able to predict both the lowest and the highest energy conformer even though the ranking of the D-pentapyranoses were not perfect. For the geometry, both PM3^{STD} and PM3^{MS} are again off by more than 0.10 Å, with the PM3^{MS} slightly better than PM3^{STD}.

Table 3.6: Relative energies (in kJ/mol) of the different $^4C_1$ conformations of pentapyranoses and the average r.m.s. deviations with respect to MP2/cc-pVTZ.

Conformer	MP2	PM3^{STD}	PM3^{MS}
α-D-xylopyranose	0.0	0.0	0.0
β-D-ribopyranose	1.0	-3.9	4.9
α-D-lyxopyranose	2.6	-2.9	2.0
α-D-arabinopyranose	3.2	-4.9	1.7
β-D-lyxopyranose	5.3	-2.4	8.3
α-D-ribopyranose	7.2	6.5	5.5
β-D-xylopyranose	9.5	-3.1	8.5
β-D-arabinopyranose	12.2	3.4	13.4
Average r.m.s.d.		7.2	2.0

3.3 Conclusion

The PM3^{MS} parameterization represents a major improvement over PM3^{STD} in correctly predicting the energies of the lowest and the highest energy conformers of the training set. Although the PM3^{MS} did not predict a perfect ranking of energies within the training set, the energy gap is quite small with respect to the reference. These small differences in energy might not be relevant when the PM3^{MS} parameters are used in molecular dynamic simulations.

Table 3.7: S_{rms} deviations (in Å) of the different $^4C_1$ conformations of pentapyranoses with respect to MP2/cc-pVTZ.

Conformer	PM3^{STD}	PM3^{MS}
α-D-xylopyranose	0.07	0.09
β-D-ribopyranose	0.19	0.15
α-D-lyxopyranose	0.10	0.11
α-D-arabinopyranose	0.10	0.10
β-D-lyxopyranose	0.13	0.11
α-D-ribopyranose	0.10	0.12
β-D-xylopyranose	0.14	0.09
β-D-arabinopyranose	0.15	0.11
Average r.m.s.d.	0.12	0.11

By using the PM3^{MS} in predicting the relative energies of monosaccharides outside the training set, it is found that it again outperforms the PM3^{STD}. Both PM3^{MS} and PM3^{STD} overestimate the overall geometry of the reference molecules by more than 0.10 Å.

Chapter 4

Free Energy Calculations: Theory

Free energy is an important quantity in the interpretation of many chemical and physical phenomena. Binding between a host and a guest molecule, solvation and diffusion phenomena, solvent effects on reactions, enzymatic reactions, protein folding, and many other properties can all be understood if the underlying free energy profiles are known. More importantly, a detailed knowledge of the free energy profile of a molecular system would allow one to predict how that particular system will behave.

Computer calculations of free energy of large molecular systems are more challenging than first thought. This has less to do with the foundations of a particular computational approach, e.g., molecular dynamics (MD) or Monte Carlo (MC) method, but more with the questions of how to adequately perform the molecular sampling technique and how reliable are the models used to represent the molecular systems, e.g. force-fields used. Early attempts of using MD or MC for configurational sampling for complex systems were reported in the early 1980s [145, 146]. Some of the issues, problems and difficulties related to sampling and the adequacy of force-fields when performing free energy calculations have been discussed [147, 148, 149, 150, 151, 152]. Even with the advent of fast computers, the exploration of the full conformational space is still too slow to perform a complete sampling required to reliably predict the free energy.

There are various methods of performing free energy calculation. These methods include free energy perturbation, umbrella sampling, thermodynamic integration, and slow growth method. In this chapter, the free energy perturbation and umbrella sampling methods are discussed since these two are the methods used in the present work.

4.1 Methods to Calculate Free Energies

The free energy of a system is usually expressed as either the Helmholtz function A, or the Gibbs function G. In thermodynamics, the Helmholtz free energy measures useful work that is obtainable from a closed system at constant temperature while the Gibbs free energy measures the same useful work that is obtainable from isothermal and isobaric systems. In molecular dynamics, the Helmholtz free energy is obtained by performing an MD simulation for a system in constant NVT condition. The Gibbs free energy, on the other hand, is determined when the MD simulation is performed under constant NPT condition.

In the discussion of the methods to compute the Helmholtz or the Gibbs free energies using MD, it is important to briefly discuss the phase space. *Phase space* is a representation of all the possible states of a system. Each possible state of the system corresponds to one unique point in the phase space. Recall from Eqs. (1.109) and (1.110) that the Hamiltonian is dependent on the position, $\mathbf{r}_i$, and momentum, $\mathbf{p}_i$, of particle i:

$$\dot{\mathbf{r}}_i = \frac{d\mathbf{r}_i}{dt} = \frac{\partial \hat{H}}{\partial \mathbf{p}_i} \tag{4.1}$$

$$\dot{\mathbf{p}}_i = \frac{d\mathbf{p}_i}{dt} = -\frac{\partial \hat{H}}{\partial \mathbf{r}_i} \tag{4.2}$$

and can be written as $\hat{H} = \hat{H}(\mathbf{r}_i, \mathbf{p}_i)$. The Hamilton equations describe the way in which the system moves through the phase space. Therefore, for a system consisting of N atoms, $6N$ values are required to define the state of the system, i.e., $3N$ positions plus $3N$ momenta of atoms. In molecular dynamics simulations, for example, the points in the phase space, which are connected in time, can be sequentially generated. In short, each point in the phase space thus corresponds to a unique configuration of the system at a given time. If the molecular dynamics simulation is performed, say in a constant NVT ensemble, then the sampling of the phase space will occur along a simulation landscape of constant temperature.

As previously mentioned, the free energy of a system in computer simulations is a difficult quantity to evaluate. This is especially true if the phase space, in particular the configuration space, of the system is separated by energy barriers that leads to computer simulations leaving the inaccessible configurations poorly sampled, or not sampled at all.

The free energy perturbation and umbrella sampling are among two of the methods designed for sampling the phase space of the system. Both methods work by restricting the region of phase space that needs to be sampled in a simulation.

4.1.1 The Free Energy Perturbation

The calculations of Helmholtz and Gibbs free energies differ only in the way in which the MD simulation is carried out, either at constant NVT (Helmholtz) or at constant NPT (Gibbs). In the following discussion, computation of free energy is described in the context of the Helmholtz free energy.

In statistical mechanics, the Helmholtz free energy for a system can be written as:

$$A = -k_B T \ln \mathrm{Q_{NVT}} \tag{4.3}$$

where Q_{NVT} is the partition function obtained under a constant NVT condition:

$$Q_{NVT} = C \int \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\frac{\hat{H}(\mathbf{r}^N, \mathbf{p}^N)}{k_B T}\right]. \tag{4.4}$$

In Eq. (4.4), $\mathbf{r}^N$ and $\mathbf{p}^N$ represents the positions and momenta of N particles, respectively, C is some constant factor, k_B is the Boltzmann's constant, T is the absolute temperature, and $\hat{H}(\mathbf{r}^N, \mathbf{p}^N)$ is the Hamiltonian of the system. The Hamiltonian equals the sum of the potential energy, $V(\mathbf{r}^N)$, which depends upon the positions of the particles, and the kinetic energy of the system, $K(\mathbf{p}^N)$, which depends upon the momenta of the particles.

Let us consider two well-defined states I and J. State I contains N particles interacting according to the Hamiltonian, $\hat{H}_I$ and state J contains N particles interacting according to the Hamiltonian, $\hat{H}_J$. The Helmholtz free energy difference, ΔA, between the two states, I and J, of the system with partition functions Q_I and Q_J, respectively, is:

$$\Delta A = A_J - A_I = -k_B T \ln \left(\frac{Q_J}{Q_I} \right). \tag{4.5}$$

Using the partition function Q_{NVT} in Eq. (4.4), Eq. (4.5) can be written as:

$$\Delta A = -k_B T \ln \left(\frac{\int \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\hat{H}_J(\mathbf{r}^N, \mathbf{p}^N)/k_B T \right]}{\int \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\hat{H}_I(\mathbf{r}^N, \mathbf{p}^N)/k_B T \right]} \right). \tag{4.6}$$

By utilizing the following equality:

$$1 = \exp\left[+\hat{H}_I(\mathbf{r}^N, \mathbf{p}^N)/k_B T \right] \exp\left[-\hat{H}_I(\mathbf{r}^N, \mathbf{p}^N)/k_B T \right] \tag{4.7}$$

and using it to multiply the numerator in Eq. (4.6), ΔA can be written in terms of an ensemble average with respect to configurations of the state I, $\langle \ldots \rangle_I$:

$$\begin{aligned} \Delta A &= -k_B T \\ &\times \ln \left(\frac{\int \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\left(\hat{H}_J - \hat{H}_I \right)/k_B T \right] \exp\left[-\hat{H}_I/k_B T \right]}{\int \int d\mathbf{r}^N d\mathbf{p}^N \exp\left[-\hat{H}_I/k_B T \right]} \right) \\ &= -k_B T \ln \left\langle \exp\left[-\left(\hat{H}_J - \hat{H}_I \right)/k_B T \right] \right\rangle_I \end{aligned} \tag{4.8}$$

where $\hat{H}_I$ and $\hat{H}_J$ represent $\hat{H}_I(\mathbf{r}^N, \mathbf{p}^N)$ and $\hat{H}_J(\mathbf{r}^N, \mathbf{p}^N)$, respectively. This method of calculating free energy differences was introduced by Zwanzig [153] in 1954.

The average in Eq. (4.8) converges rapidly only when the difference between the energies of the two states I and J is very small, i.e., of the order of $k_B T$. In practice, however, it is practically impossible to carry out enough sampling to properly evaluate the ΔA in Eq. (4.8). The solution is to break down the problem into smaller ones by introducing a *perturbation parameter*, λ, into the Hamilton equations of motion. In this situation, the Hamiltonian becomes a function of not only the positions and momenta of the particles, but also a function of the parameter λ: $\hat{H} = \hat{H}(\mathbf{r}^N, \mathbf{p}^N, \lambda)$.

Several λ-coupling schemes have been suggested. The simplest among the coupling schemes that define the intermediate states for the transition between the states I and J is by a direct linear interpolation between the two states:

$$\hat{H}(\mathbf{r}^N, \mathbf{p}^N, \lambda) = (1 - \lambda)\hat{H}_I(\mathbf{r}^N, \mathbf{p}^N) - \lambda \hat{H}_J(\mathbf{r}^N, \mathbf{p}^N). \tag{4.9}$$

Another one is a straightforward extension of Eq. (4.9), resulting in a non-linear scheme:

$$\hat{H}(\mathbf{r}^N, \mathbf{p}^N, \lambda) = (1 - \lambda)^n \hat{H}_I(\mathbf{r}^N, \mathbf{p}^N) - \lambda^n \hat{H}_J(\mathbf{r}^N, \mathbf{p}^N). \tag{4.10}$$

where n is some exponent. In other coupling schemes, the force-field parameters are scaled linearly instead of the Hamiltonian:

$$p_\lambda = (1 - \lambda)p_I - \lambda p_J \tag{4.11}$$

where p is some force-field parameter.

Suppose that only the potential energy terms, $V_I(\mathbf{r}^N)$ and $V_J(\mathbf{r}^N)$, differ and that the kinetic energy parts of the Hamiltonian are the same, i.e., the masses and the numbers of particles are identical. Then, the kinetic energy terms would cancel out and Eq. (4.8) can also be written as a function of only the potential energy terms, $V_I(\mathbf{r}^N)$ and $V_J(\mathbf{r}^N)$:

$$\Delta A = -k_B T \ln \left\langle \exp\left(-\frac{V_J(\mathbf{r}^N) - V_I(\mathbf{r}^N)}{k_B T} \right) \right\rangle_I \tag{4.12}$$

where the angled brackets denote the average value of the enclosed quantity for each configuration in the trajectory. This average value is calculated over a simulation run for state I, in which the potential energy for state J is also calculated for each configuration of the system.

In practice, the actual computation of the free energy utilizes the perturbation parameter λ that is varied incrementally between 0 and 1. It is normally asserted that the kinetic energy contribution to the free energy is zero. Thus, the free energy difference between the two *intermediate states* i and j with different values of the perturbation parameter λ_i and λ_j, respectively, is calculated in the following manner:

$$\Delta A = A_j - A_i = -k_B T \ln \left\langle \exp\left(-\frac{V(\mathbf{r}^N,\lambda_j) - V(\mathbf{r}^N,\lambda_i)}{k_B T}\right)\right\rangle_{\lambda_i} \tag{4.13}$$

It should be noted in Eq. (4.13) that $V_I(\mathbf{r}^N) = V(\mathbf{r}^N, \lambda = 0)$ and that $V_J(\mathbf{r}^N) = V(\mathbf{r}^N, \lambda = 1)$. The total free energy difference is the sum of the individual free energy differences between the intermediate states for the transition from state I to state J:

$$\Delta A_{I\rightarrow J} = \sum_{i=1}^{N_w} \Delta A_{\lambda_{(i-1)}\rightarrow\lambda_{(i)}} \tag{4.14}$$

where N_w is the total number of intermediate states or windows and the states $i = 1$ and $i = N_w$ refer to the end states, I and J, respectively. Alternatively, ΔA can be written as:

$$\Delta A_{I\rightarrow J} = \sum_{\lambda=0}^{\lambda=1} (A_{\lambda+\delta\lambda} - A_\lambda) \tag{4.15}$$

where $\lambda = 0$ and $\lambda = 1$ refer to the end states, I and J, respectively, and $\delta\lambda$ is an increment value. The free energy calculation described in Eq. (4.13) and Eqs. (4.14)–(4.15) is commonly refered to as the *free energy perturbation* (FEP) method.

A major advantage of the FEP method is that the perturbation can be done on a non-physical process. This makes it possible to calculate the free energy differences by *alchemical perturbation*, in which one molecule is transformed or "mutated" onto another by altering the number and the corresponding atom types involved during the simulation. For example, the relative hydration free energies of methanol and ethanol molecules can be calculated by performing a simulation in which the methanol molecule is transformed into ethanol.

Since the perturbation in the FEP method does not necessarily have to correspond to a physically realizable process, another useful application of the FEP technique is in the process of calculating the relative binding energies between two subsystems. In calculating relative binding energies using the FEP method, an interconversion pathway in the form of the *thermodynamic cycle*, like the one shown in Figure (4.1), is usually defined. In this scheme, for example, S and S' represent a substrate and a modified substrate, respectively,

$$\begin{tikzcd}
S + (BM) \arrow[r, "\Delta A_S"] \arrow[d, "\Delta A_1"] & S(BM) \arrow[d, "\Delta A_2"] \\
S' + (BM) \arrow[r, "\Delta A_{S'}"] & S'(BM)
\end{tikzcd}$$

Figure 4.1: The thermodynamic cycle.

BM represents a biomolecule, and the ΔAs represent free energy changes for the indicated processes. Because the free energy is a thermodynamic state function, the free energy difference is independent of the path over which the change occurs and depends only on the initial and final states of the system. This means that the sum of the individual ΔA in Figure (4.1) is zero.

$$\Delta\Delta A = \Delta A_S + \Delta A_2 - \Delta A_{S'} - \Delta A_1 = 0 \tag{4.16}$$

Therefore, the relative binding of S and S' to BM can be calculated from the non-physical processes represented by vertical arrows:

$$\begin{aligned} \Delta\Delta A_{bind} &= \Delta A_{S'} - \Delta A_S \\ &= \Delta A_2 - \Delta A_1. \end{aligned} \tag{4.17}$$

The combination of FEP and thermodynamic cycle can be formulated for many other similar problems of interest. The key point is to formulate the interconversion pathways in such a way that they would correctly represent the actual physical process or problem being modeled.

4.1.2 The Umbrella Sampling

Another way of calculating free energy is by considering changes in geometry or conformation of the system. Conformational changes in the system would be defined by a profile of the free energy with respect to the conformational variables, which in turn are defined by a conformational constraint or restraint on the system. The resulting free energy map along some reaction coordinate or coordinates is known as the *potential of mean force* or PMF.

Unlike the free energy calculations in the FEP method which are often along non-physical pathways, the PMF is usually calculated for a physically achievable process. Thus, PMF is useful for elucidating reaction mechanisms such as an enzymatic process.

The PMF $\omega(\chi)$ along some PMF coordinate χ is defined from the average distribution function $\langle\rho(\chi)\rangle$:

$$\omega(\chi) = -k_B T \ln \langle\rho(\chi)\rangle + C \tag{4.18}$$

where C is some arbitrary constant. The average distribution function, $\langle\rho(\chi)\rangle$, can be expressed as:

$$\langle\rho(\chi)\rangle = \frac{\int d\mathbf{r}^N \delta(\chi'(\mathbf{r}^N) - \chi) \exp\left[-V(\mathbf{r}^N)/k_B T\right]}{\int d\mathbf{r}^N \exp\left[-V(\mathbf{r}^N)/k_B T\right]}. \tag{4.19}$$

In Eq. (4.19), the assumption is made that the chosen PMF coordinate χ is a geometrical coordinate, $\chi(\mathbf{r}^N)$, which is independent of momentum of the particle. It must be noted, however, that χ in general can be assigned any other functionality. The Dirac delta function, $\delta(\chi'(\mathbf{r}^N) - \chi)$, ensures that only those atomic coordinates $\mathbf{r}^N$ that give the reference value of the PMF coordinate χ, are selected.

For processes with an energy barrier higher than $k_B T$, the average distribution function $\langle\rho(\chi)\rangle$ cannot be computed properly by a straight molecular dynamics simulation due to low sampling in higher-energy configurations. A solution to solve the sampling problem is to use a technique that encourages energy barrier crossing. One such technique is the *umbrella sampling* method.

The umbrella sampling method was first suggested by Torrie and Valleau in 1977 [154]. By assuming that χ is a geometrical coordinate, in the umbrella sampling method an artificial biasing window potential function, $V_{umb}(\chi)$, which can take a harmonic function of the form

$$V_{umb}(\chi)_i = \frac{1}{2} k_{umb} (\chi - \chi_i)^2 \tag{4.20}$$

is added to the standard potential energy function. In Eq. (4.20), i is a window or an intermediate state, χ is the reference value of the PMF coordinate, χ_i is the reference value of the coordinate for the window i, and k_{umb} is the force constant for the potential. The umbrella potentials, $V_{umb}(\chi)$, center the sampling in the different overlapping windows of χ.

In the umbrella sampling technique at each window i, the unbiased PMF, $\omega(\chi)_i^{unbiased}$ in Eq. (4.21), is expressed in terms of the biased average distribution function, $\langle\rho(\chi)\rangle_i^{biased}$, the biasing potential function, $V_{umb}(\chi)_i$, and the undetermined constant, F_i, which represents the free energy associated with the biasing window potential:

$$\omega(\chi)_i^{unbiased} = -k_B T \ln \langle\rho(\chi)\rangle_i^{biased} - V_{umb}(\chi)_i + F_i \tag{4.21}$$

The final estimate for the full PMF $\omega(\chi)$ is constructed by recombining the separate distributions from Eq. (4.21). An efficient procedure for achieving this is by using a strategy called the *Weighted Histogram Analysis Method* or WHAM [155]. The WHAM method determines the optimal estimate for the unbiased average distribution function, $\langle\rho(\chi)\rangle$, from the biased average distribution function for each window, $\langle\rho(\chi)\rangle_i^{biased}$. In particular, the total unbiased distribution, $\langle\rho(\chi)\rangle$, is calculated as a weighted sum of unbiased window distribution functions, $\langle\rho(\chi)\rangle_i^{unbiased}$:

$$\langle\rho(\chi)\rangle = \sum_{i=1}^{N_w} \langle\rho(\chi)\rangle_i^{unbiased} \left(\frac{n_i \exp\left[-(V_{umb}(\chi)_i - F_i)/(k_B T)\right]}{\sum_{i=1}^{N_w} n_i \exp\left[-(V_{umb}(\chi)_i - F_i)/(k_B T)\right]} \right), \tag{4.22}$$

or alternatively, $\langle\rho(\chi)\rangle$ can be expressed in terms of the known biased distribution functions, $\langle\rho(\chi)\rangle_i^{biased}$:

$$\langle\rho(\chi)\rangle = \frac{\sum_{i=1}^{N_w} n_i \langle\rho(\chi)\rangle_i^{biased}}{\sum_{i=1}^{N_w} n_i \exp\left[-(V_{umb}(\chi)_i - F_i)/(k_B T)\right]}. \tag{4.23}$$

In Eqs. (4.22) and (4.23), n_i is the number of independent data points employed for the generation of the distribution for window i, N_w is the total number of simulation windows, and F_i is the free energy from the simulation window i given by the following expression:

$$F_i = -k_B T \ln \int d\chi \, \langle\rho(\chi)\rangle \exp\left[-V_{umb}(\chi)_i/(k_B T)\right] \tag{4.24}$$

The distribution function, $\langle\rho(\chi)\rangle$, and the N_w free energies, F_i, are initially unknown and must be solved iteratively to self consistency. The procedure used assumes a set of guess values (usually zero) for the free energies of each window. These guess values are then used to calculate $\langle\rho(\chi)\rangle$ for the complete range of χ from Eq. (4.23). The estimated $\langle\rho(\chi)\rangle$ is then used to determine the free energies of each window from Eq. (4.24). This procedure is repeated until the values both of $\langle\rho(\chi)\rangle$ and of the set of F_i no longer change. In order to obtain accurate results using this self-consistent procedure, the histograms corresponding to neighboring distribution functions must overlap to a reasonable extent.

In practice, several considerations must be made when performing simulations using the umbrella sampling method. First is the choice of the reaction coordinate since PMF depends on the chosen reaction coordinate. The reaction coordinate may be defined in a way that it directly corresponds to the actual physical process of the system being modeled. This could involve changes in geometrical coordinates such as bond lengths, bond angles, or dihedral angles. Second is the number of simulations to be performed. This corresponds to N_w or the number of windows to obtain the PMF. In general, the number of simulation windows covering the whole range on the reaction coordinate must be sufficient. The data points must be sampled in such a way that there are good overlaps between neighboring windows. Third, which is related to the previous one, is the choice of the biasing potential (strong or weak). Choosing a strong biasing potential could mean more simulations must be performed to cover the whole range of the reaction coordinate. Choosing a weak biasing potential, on the other hand, allows for fewer simulations but may not be enough to ensure that adequate sampling has been done to capture processes in the high energy or unfavorable regions of the system being modeled. Fourth is the choice of the simulation condition, e.g. constant NVT or constant NPT, to determine which free energy is being calculated, i.e., Helmholtz or Gibbs free energy. Finally, a careful examination of the histograms must be made. There should be no major shift from the expected value of the PMF coordinate at the current window. The histograms must also be smooth since noisy histograms can lead to inaccurate PMF.

4.2 Conclusion

Two techniques for calculating free energy have been presented in this chapter. The free energy perturbation and the umbrella sampling methods provide two different ways of calculating the free energies. The free energy perturbation method computes free energy via non-physical process. The umbrella sampling method, on the other hand, calculates the free energy using physically achievable processes.

Computing free energy is a challenging task in computer simulation. Using MD or MC simulation methods alone cannot adequately and efficiently sample the entire phase space given that computer resources are limited by the size of the system being studied. The other consideration is the amount of time that can be given for the simulation. Sampling techniques, such as those described in this chapter, are often designed to achieve an efficient and more complete exploration of the phase space.

Chapter 5

Cancer Cells, Chemotherapy Drugs and Microtubules

Cancer affects the lives of many people. It is a debilitating disease that destroys the way people live their lives both physically and emotionally. Cancer, if not caught at its early stage, can lead to the shortening of the life of a person.

Cancer is a group of many related diseases related to the basic building block of life, the *cell.* Normal cells grow and divide, and then eventually die. Cancer occurs when some cells grow at a very fast rate and divide out of control. Cancer cells usually group or clump together to form *tumors.* A growing tumor can destroy the normal cells nearby, therefore destroying the surrounding healthy tissues in the process.

Cancer spreads via cancer cells breaking away from the original tumor and then traveling to other areas of the human body, where they can keep on growing and form new tumors. The spread of a tumor to a different place in the human body is called *metastasis.*

Cancer can be treated with surgery, radiation therapy, chemotherapy, or combinations of these treatments. The choice of treatment will depend on the kind of abnormal cells causing the cancer and the stage of the tumor. Surgery is performed by surgically removing as many cancer cells possible, with the possibility of also removing healthy or normal cells in the process, to ensure that the cancer cells are removed. Surgery is an invasive procedure in which an incision is made to access the inside of the body where the tumor to be removed is found.

Radiation therapy, also known as radiotherapy, uses ionizing radiation or high-energy waves, e.g. X-ray, to kill cancer cells and shrink the tumors. Radiation therapy destroys cancer cells in the target tissue by damaging their genetic material. Cancer cells damaged in this way cannot continue to grow and divide. Although most normal cells in the vicinity of the cancer cells where radiation is applied are also damaged, the normal cells can recover from the effects of the radiation and function properly. The application of radiation depends on several factors such as the type of cancer, location of the cancer, how far into the body the radiation needs to go, the general health of the patient including the medical history, just to name a few.

Chemotherapy uses anti-cancer or *cytotoxic* drugs to treat cancer by stopping the cancer cells from rapidly dividing and reproducing, and eventually leads to their destruction. In chemotherapy, the anti-cancer or chemotherapy drugs are usually specific in the sense that they target and damage specific cancer cells. It is one of the reasons why a combination of chemotherapy drugs is usually given to a patient who developed different types of cancer. The chemotherapy treatment usually can last for several weeks or even months. There are a number of chemotherapy drugs, available on the market, designed to combat a specific type or types of cancer. Table 5.1 shows some of the drugs, their brand names and the types of cancer they are administered for. The possible side effects of taking these drugs may include lowered resistance to infection caused by reduced production of white blood cells, bruising or bleeding caused by reduced production of platelets, anemia or low number of red blood cells, sore mouths and ulcers, constipation, diarrhea, nausea and vomiting, loss of appetite, dizziness and blurred vision, skin pigmentation, hair loss, and tiredness.

Chemotherapy drugs can be taken orally as a pill or given through intravenous (IV) line, where directly they enter the bloodstream. Since these drugs are carried in the blood, practically nearly all cancer cells anywhere in the body can be reached and targeted. However, healthy and normal cells are also exposed to drug treatment. Healthy cells in the human body that are specially sensitive to chemotherapy drugs include the bone marrow, the digestive system, the lining of the mouth, and hair follicles. Normal cells also absorb

Table 5.1: Some commercially available chemotherapy drugs.

Chemotherapy Drug	Brand Name	Type of Cancer
Busulfan	Busilvex®, Myleran®	chronic myeloid leukemia
Capecitabine	Xeloda®	breast, advanced bowel cancer
Carmustine	BiCNU®	lymphomas, myeloma, brain tumors
Chlorambucil	Leukeran®	chronic lymphocytic leukemia, low-grade non-Hodgkin lymphoma, Hodgkin lymphoma, Waldenstrom's macroglobulinaemia
Cisplatin		testicular, bladder, lung, esophagus, stomach, ovarian
Docetaxel	Taxotere®	breast, prostate, lung
Etoposide	Etopophos®, Vepesid®	lung, ovarian testicular
Fludarabine	Fludara®	chronic lymphocytic leukemia
Fluorouracil	5FU	bowel, breast, stomach, esophagus
Gemcitabine	Gemzar®	lung, pancreatic, bladder, breast
Hydroxycarbamide	Hydrea®	chronic myeloid leukemia, cancer of the cervix
Irinotecan	Campto®	bowel
Mercaptopurine	Puri-Nethol®	acute leukemia
Oxaliplatin	Eloxatin®	cancer of the large bowel
Paclitaxel	Taxol®	breast, ovarian, lung
Pemetrexed	Alimta®	lung
Temozolomide	Temodal®	brain tumor
Vinblastine	Velbe®	leukemia, lymphoma, breast, lung
Vincristine	Oncovin®	leukemia, lymphoma, breast, lung
Vindesine	Eldisine®	leukemia, lymphoma, melanoma breast, lung
Vinorelbine	Navelbine®	breast, lung

the chemotherapy drugs and this can cause unpleasant side effects. The side effects caused by the absorption of chemotherapy drugs by the healthy cells, however, may be temporary if the drug is given in the correct dosage.

Chemotherapy drugs have a relatively high rate of failure. While these drugs usually kill only specific types of cancer cells within a tumor, they may be too toxic and effectively kill normal cells. In the treatment of cancer in patients, combinations of chemotherapy drugs are usually given. The goal is to achieve minimum damage to normal cells and enhanced cytotoxic effects on cancer cells.

Cancer cells can adapt to the toxic environment, posed by chemotherapy drugs, via mutation. Eventually, they become resistant to the chemotherapy drug. Cancer cells also adapt by invading and utilizing biological mechanisms of normal cells to assure their continued existence. Understanding the specific cellular process and controlling the overproduction of certain cellular components involved in the rapid growth of cancer cells play a critical role in the development and design of effective chemotherapy drugs.

Many chemotherapy drugs work by targeting microtubules and, consequently, by either promoting or inhibiting their polymerization. In the following sections, a brief discussion of the cell is followed by a description of the cytoskeleton, the cell cycle, and the roles that microtubules play in cell division.

5.1 The Cell

The cell is the basic organizational unit of all living organisms, which can be *unicellular* or single cell organisms (e.g. most bacteria) or *multicellular* (e.g. humans). Cells can be classified into two major types: *prokaryotic* or *eukaryotic* cells. The major differences between prokaryotic cells and eukaryotic cells are shown in Table 5.2. Prokaryotic cells are approximately 10 times smaller than eukaryotic cells. Prokaryotic cells lack the nucleus and the genetic materials floats freely in the cytoplasm. Prokaryotic cells also lack all the other organelles except for the cell walls and the ribosomes, which are smaller compared to eukaryotic cells.

A cell has a membrane that envelopes the cell, separating and protect-

Table 5.2: Comparison between prokaryotic and eukaryotic cells.

	Prokaryotic cells	Eukaryotic cell
Organization	unicellular	multicellular
Cell division	binary fission	mitosis
Cell movement	flagella made of flagellin	flagella and cilia made of tubulin
Size	~1-10 μm	~10-100 μm
Nucleus	none	nucleus with double membrane
DNA	usually circular	linear, e.g. chromosomes
RNA synthesis	in cytoplasm	inside the nucleus
Protein synthesis	in cytoplasm	in cytoplasm
Ribosomes	70S	80S
Mitochondria	absent	present
Chloroplast	none	present in algae and plants

ing it from the surrounding environment. The cell membrane or the *plasma membrane* is a selective and permeable material that regulates what goes in and out of the cell. The plasma membrane is mostly made up of *phospholipid bilayer*. Inside the plasma membrane is the cytoplasm. The cytoplasm is a semi-transparent fluid that takes up most of the volume of the cell. It is the part of the cell where most of the cellular activities are taking place. The cytoplasm is the collective term used to refer to the *cytosol* plus the *organelles* and the nutrients and secretory products of the cell suspended within the cytosol. Organelles are metabolic machines of the cell. Table 5.3 shows a listing and functions of membrane-bound organelles found in eukaryotic cells. The cytosol, which made up of mostly water, is full of proteins that control cell metabolism including signal transduction pathways, glycolysis, intracellular receptors, and transcription factors.

Table 5.3: The organelles in eukaryotic cells and their function.

Organelle	Functions
Cell nucleus	contains the chromosomes; the site of DNA replication and RNA synthesis
Centrosome	produces and organizes the microtubules of the cell
Mitochondrium	generates energy in the cell via respiration
Chloroplast	generates energy in the cell via photosynthesis
Ribosomes	serve as the sites for proteins synthesis and production
Lysosomes and peroxisomes	contain destructive enzymes for destroying the cell; also known as "suicide bags"
Endoplasmic reticulum and Golgi apparatus	manage the transport of molecules for specific destinations
Vacuoles	store food and waste

5.2 The Cytoskeleton and the Filamentous Protein Assemblies

The *cytoskeleton* is contained in the cytosol of the cell. The cytoskeleton is a dynamic cellular structure that establishes and maintains the shape of the cell, provides mechanical structure, enables cellular motion, provides the means for intracellular transport of organelles, and plays an important role in cell division. The cytoskeleton is made up of three classes of filamentous protein assemblies: the *microfilaments*, the *intermediate filaments*, and *microtubules.*

The *microfilaments* are made of actin. There are two forms of actin, *F-actin* and *G-actin.* The filamentous actin, *F-actin,* is the thinnest (~8 nm in diameter) of all the cytoskeletal filaments and is the most important form of actin contained within the cell. *F-actin* is formed through the polymerization reaction involving the globular actin monomer, *G-actin.* The polymerization of actin happens by a series of nucleation-elongation processes in which the elongation proceeds by the addition of the monomer to a seed nucleus [156, 157, 158, 159]. The polymerization is a cooperative process in which the filaments can form via polymerization when the concentration of actin exceeds a critical concentration, typically in the range of 0.1 to 2 μM [160].

Actin provides a dynamic component of the cytoskeleton [161]. Actin filaments form a band just beneath the plasma membrane providing mechanical strength to the cell and link cell surface receptors or *transmembrane proteins* to the cytoplasmic proteins. The actin filaments also generate locomotion in cells, e.g. in amoeba and in white blood cells. They also interact with the filaments in the skeletal muscle fibers to provide the force of muscular contraction.

The *intermediate filaments* (IFs) are microscopic protein ropes in the cytoskeleton of all eukaryotic cells. An IF has a diameter size of about 10 nm. They are thicker than actin filaments and thinner than microtubules. These structures are diverse and are formed by members of a family of related proteins. Several types of intermediate filaments can be categorized into the following: *epithelial keratins* in epithelial cells; *trichocytic keratins* which form hair, nails and horns; *nuclear lamins* which stabilize the inner membrane of the nuclear envelope by forming a meshwork; *neurofilaments* which strengthen the long axon of neurons, and *vimentins* which support the cellular membrane and keep some organelles held in fixed places within the cytoplasm.

5.3 Microtubules and the Cell Cycle

In eukaryotes, cells replicate via the *cell cycle* as shown in Figure 5.1. The cell cycle can be divided into two periods – the *interphase*, which is the longest part of the cell cycle, and the *mitotic* phase.

The interphase consists of three distinct phases in the following order: the G_1 or *Gap* 1 phase, the S or *Synthesis* phase, and the G_2 or *Gap* 2 phase. The G_1 phase, immediately following the end of a previous mitotic phase, is marked by an increase in the cell size and a high rate of biosynthesis of RNA and various enzymes needed for DNA replication in the S phase. In the S phase, DNA replication occurs. In the G_2 phase, the cell continues to grow and biosynthesizes more proteins including the production of microtubules, which are required during the mitotic phase.

The mitotic or M phase is the period in the cell cycle in which protein production and cell growth stop. At this stage, nuclear division or *karyoki-*

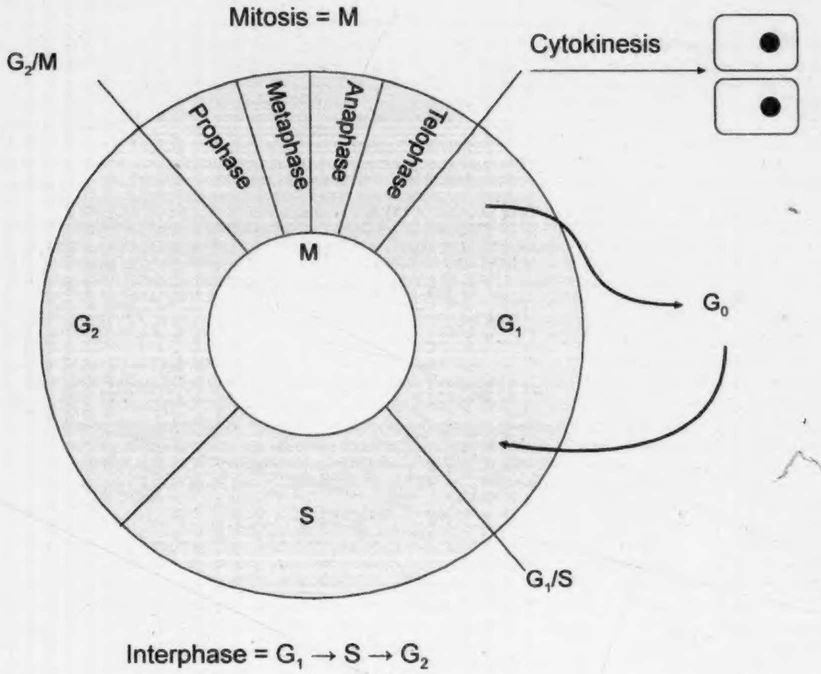

Mitosis = M
G2/M
Cytokinesis
Prophase
Metaphase
Anaphase
Telophase
M
G2
G1
G0
S
G1/S
Interphase = G1 → S → G2

Figure 5.1: The cell cycle.

nesis and cytoplasmic division or *cytokinesis* occur. Mitosis is the process in which the nucleus of the cell divides where the duplicated chromosomes separate to form two genetically identical daughter nuclei. Nuclear cell division during mitosis occurs in several stages: *prophase*, *metaphase*, *anaphase*, and *telophase*. During *prophase*, the nuclear membrane disintegrates and the nucleolus disappears. A nucleolus is a knot of chromatin, which is an active form of DNA. As prophase continues, chromosomal condensation begins as evidenced by the shortening and thickening of the chromosomes. Mitotic spindles begin to form between the poles and kinetochores, which are protein structures on the chromosome, begin to mature and attach to the spindles. Mitotic spindles are made up of microtubules. During *early metaphase*, chromosomes attach to the mitotic spindle and line up along the metaphase plate at the equator of the cell. During *late metaphase*, the centromeres of each chromosome divides (a centromere is a region in the middle of the chromosome that serves as the point of attachment for spindle fibers). During *anaphase*, the daughter chromosomes move to opposite poles of the cell in which the daughter chromosomes are assisted by spindle fibers. The kinetochore microtubules shorten, assisting in the separation of the chromosomes to opposite poles while the polar microtubules lengthen and elongate the cell. During *telophase*, the polar microtubules continue to elongate. At this stage, the separation of the chromosomes is complete. The nuclear membranes form, the nucleoli appear, and the chromosomes de-condense leading to the formation of new nuclei. Then the cell enters cytokinesis in which the cell divides.

At the end of the M phase, the cell either can enter the G_1 phase and repeat the cycle or leave the cycle and enter a resting stage G_0 or *Gap* 0 phase. In the G_0 phase or "post-mitotic" stage, the cell may rest temporarily, or permanently quit dividing. Most of the time, cells that enter the G_0 phase are terminally differentiated. For example, a neuron that has reached the end of its development enters the G_0 permanently and would not divide anymore. Differentiated cells do not reenter the cell cycle but instead carry out their function until they die. Cells that temporarily enter the G_0 phase can reenter the G_1 of the cell cycle via stimulation and then proceed on to another round of S, G_2, and M phases. For example, a lymphocyte (a type of white blood cell in human blood) that is in G_0 phase can be stimulated by an antigen to reenter the cell cycle. (An *antigen* is a substance that is capable of causing the production of an *antibody*, a specialized immune proteins.) Cancer cells do not enter the G_0 phase and the cell cycle is repeated indefinitely leading to their continued growth and production.

The cell cycle also has two main checkpoints for monitoring and regulating the progress of the cell cycle: the G_1/S checkpoint and the G_2/M checkpoint. These checkpoints are necessary for verifying accurately if the processes at each phase of the cycle has been completed before proceeding to the next phase in the cell cycle. The G_1/S checkpoint is where a decision is made whether the cell should divide, postpone division, or enter the resting stage. Most cells stop at the G_1/S checkpoint and enter the G_0 phase. The G_2/M checkpoint ensures that the cell is ready for mitosis. This checkpoint also prevents damaged DNA from being transmitted to daughter cells by arresting the cell cycle prior to mitosis.

5.4 Microtubules, Tubulin, and Colchicine

5.4.1 Microtubules

Microtubules (MTs) are filamentous intracellular structures in all eukaryotic cells that are responsible for moving vesicles, granules, organelles like mitochondria, and chromosomes. They are important in mitosis or nuclear cell division, organization of intracellular structure, and intracellular transport, as well as ciliary and flagellar motility. A microtubule is a straight, hollow filament of cylindrical nature with about 15 nm in the inner diameter and 25 nm in the outer diameter. A microtubule is formed via polymerization of the α/β-tubulin heterodimer. Figure 5.2 shows typical structures of microtubule. The microtubule in Figure 5.2A forms a perfect helical spiral while the microtubule in Figure 5.2B shows a structural discontinuity.

5.4.2 Tubulin isoforms and their 3D Models

Homologous protein families, such as tubulin, are collectively known as isoforms and have amino acid sequences that diverged as a result of accumulated mutations since their separation by speciation events [162]. The resulting vari-

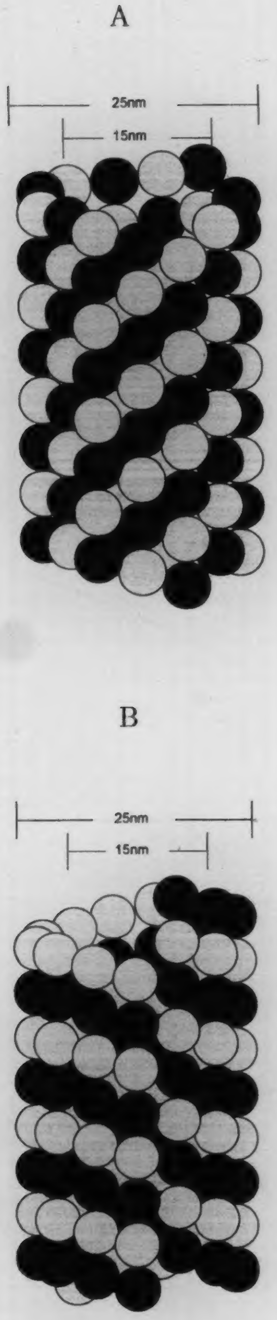

A
25nm
15nm
B
25nm
15nm

Figure 5.2: Typical structures of microtubule.

ations in sequence can be neutral (when they are irrelevant to the process of natural selection) or essential (when they adapt the function of a protein to a given selective pressure).

A model representation of α/β-tubulin heterodimer is given in Figure 5.3. At the molecular level, the role of tubulin is extremely complex and seems to be related to structural variations observed between α- and β-isoforms [163]. The existence and distribution of tubulin isoforms provides a link to their structure and their role in the polymerization and stability of microtubules. It is clear that much of the tubulin surface is invariant, however those substitutions that do occur are clustered at positions that comprise the longitudinal interface between protofilaments [164]. This observation implies that there must be a contribution from the inter-dimer interface, between protofilaments, that is key to the understanding of the properties that each isoform contributes to microtubule stability.

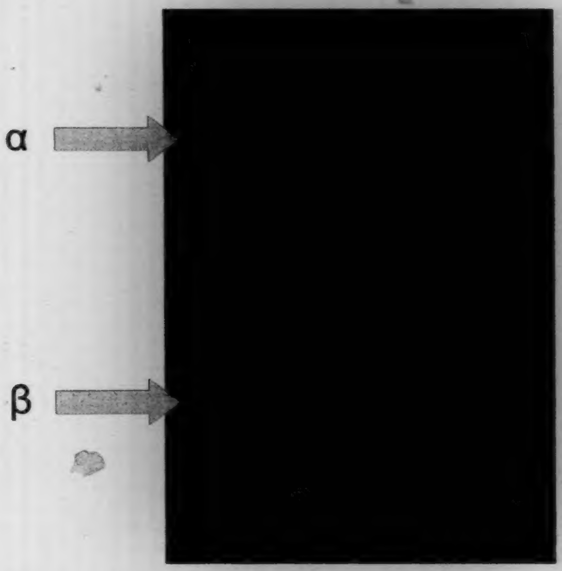


Figure 5.3: Model representation of α/β-tubulin heterodimer.

Isoform composition has previously been recognized as having a demonstrable effect on microtubule assembly kinetics [165, 166] whereby small differences in the binding energies and chemical affinities of different tubulin isoforms surprisingly translate into significant deviations in the growth rates and catastrophe frequencies. Short-range interactions have been studied by calculating the energy of protofilament-protofilament interactions [167].

Cells, especially cancer cells, are capable of altering the expression of each tubulin isoform (encoded by different genes) in response to external conditions that affect microtubule stability. There are several examples of this response, most recent is the over-expression of β-tubulin isoform III (βIII) following exposure to microtubule stabilizing agents such as paclitaxel [168, 169, 170, 171]. Table 5.4 identifies the distribution of β-tubulin isoforms in normal human cells. Current anti-tubulin drugs bind to all of these isoforms, having only slight preference for one over another [172, 173]. For example, the vinca alkaloids bind best to βII [172], providing an explanation as to their efficacy in leukemia and Hodgkins lymphoma, since these cancerous cells express βII while normal lymphocytes do not [174]. It is worth stressing that cancerous cells seem to express a variety of tubulin isoforms, and are not limited to those expressed in the non-cancerous cells from which they are derived [175]. Therefore, a drug that is highly specific for an isoform that is found within a cancerous cell could preferentially affect only those cells, while not harming significantly non-cancerous cells.

To examine the molecular properties of tubulin isoforms and the effect that they have on microtubule dynamics and drug interactions, an earlier search [164] of both the SWISS-PROT and Entrez protein databases [176] has been performed. A total of seven human α-tubulin isoforms and eight β-tubulin isoforms have been identified. Following the identification of 83 individual protein sequences, corresponding only to β-tubulin, a ClustalW alignment was performed [177]. The alignments between the isoforms of β-tubulin were unambiguous due to the highly conserved amino acid sequences between these proteins. This alignment resulted in the filtering of both duplicates and fragmentary sequences and produced a final set of 23 unique sequences, from which eight distinct sub-types were classified, generally based on their overall amino acid sequence and specifically their carboxy terminal tail sequence [178, 179, 180, 181, 182]. Of the eight subtypes, βI (gi: 29788785)

Table 5.4: Tissue distribution of β-tubulin isoforms in normal cells.

Isoform	Organ Expression	Cellular Expression
βI	Constitutive	Most cells
βII	Brain, nerves, muscle; rare elsewhere	Restricted to particular cell types
βIII	Brain	Neurons only
	Testis	Sertoli cells
	Colon (very slight amounts)	Epithelial cells only
βIVa	Brain only	Neurons and glia
βIVb	Constitutive (not as widespread as βI)	High in ciliated cells, lower in others
βV	Unknown	Unknown
βVI	Blood, bone marrow, spleen	Erythroid cells, platelets
βVII	Brain	Unknown

contained two additional isoforms (gi: 18088719, 338695). The βII isoform contained additional two sequences (gi: 4507729, 29788768) that carried differences in their carboxy terminal tail, however, class βIIa may actually correspond to a pseudo-gene [178]. In addition to the two βII tubulin genes, three additional proteins (gi: 27227551, 49456871, 7441369) that carry minor substitutions within the coding sequence have also been identified. The βIII tubulin isoform (gi: 50592996) also contained two additional proteins (gi: 62897639, 1297274) which differ only by a single amino acid substitution. Two-class βIV tubulin genes (gi:21361322, 135470) differ only in their carboxy terminal tail sequences. The class βV (gi: 14210536) and βVI (gi: 62903515) are unique in their sequences. The class βVII tubulin gene family (gi: 55770868) contains additional two proteins (gi: 1857526, 12643363) that show slightly greater sequence variability than any of the other β-tubulin isoforms. Finally, the βVIII tubulin gene (gi: 42558279) is also unique in its sequence and has yet to be officially classified [165].

The presence of both numerous tubulin structures and multiple tubulin isoform sequences offers a unique opportunity to apply homology modeling and create a library of human β-tubulin isoforms, from which their key biochemical characteristics can be determined. Following the solution of the three-dimensional structure of a protein, it becomes possible to use homology modeling to predict the structure of a protein that has a similar sequence [183]. Homology modeling utilizes several structural motifs from template proteins and pieces them together to form a final model. A scoring function assesses both the sequence identity between the target sequence and template and the overall quality of the template that is being considered. The scores are ranked and the fold with the best score is assumed to be the one adopted by the target sequence [184]. In a similar manner, homology modeling on the complete set of human β-tubulin sequences [164] has been performed.

Currently, a total of five unique structures, upon which homology models of β-tubulin can be constructed, are available in the Brookhaven Protein Databank. The first, and most obvious choice was the initial structure of the α/β-tubulin heterodimer (PDB identifier 1TUB) proposed by Nogales *et al* [185]. However, this structure was produced at a comparatively low resolution and contains numerous omissions and misalignments due to difficulties in density fitting. A refinement of the 1TUB structure (PDB identifier 1JFF) was subsequently deposited with a slightly better resolution at 3.5 Å [186]. Along with increased resolution, the 1JFF structure also addressed a number of misalignment errors within the 1TUB structure and therefore makes a substantially better choice for a homology modeling template. Two additional structures of the tubulin heterodimer that are complexed with the stathmin-like domain, RB3-SLD, in the presence of colchicine (PDB identifier 1SA0) and vinblastine (PDB identifier 1Z20) are also available [187, 1]. The 1SA0 structure was solved at 3.5 Å, while the 1Z20 structure was refined to only 4.1 Å. While the resolution of 1SA0 is slightly better than that in 1JFF, this structure contains a nine-residue gap in β-tubulin, ranging from Ser277 to Leu286. Both 1JFF and 1SA0 were chosen to be used as templates for modeling the human β-tubulin isoforms. Because the sequences within the β-tubulin family are more similar to each other than to the other tubulin isoforms, it is reasonable to believe that any given sequence should produce a structure very similar to another member of a given family. Further support for this comes from the published structures of Nogales *et al.* [185] and Lowe *et al.* [186], which are of a porcine sequence that were fit to structural data from an inhomogeneous bovine sample. Accordingly, by substituting appropriate amino acid side chains and properly adjusting other residues to accommodate insertions and deletions in the sequence, these crystallographic structures can be used as a framework to produce, with a high degree of confidence, model structures with different sequences.

To build the β-tubulin models the Modeller program (version 6v2) [188, 189], which uses alignment of the sequences with known related structures, has been used to obtain spatial restraints that the output structure must satisfy. Additional restraints derived from statistical studies of representative protein and chemical structures were also used to ensure a physically plausible result. Missing regions were predicted by simulated annealing using a molecular mechanics model. Although the Modeller program provides many options for model refinement and tuning, only the default parameters were used, as simple visual inspection of some of the output structures suggested that reasonable models were being produced. The quality of the resulting models was then investigated using two software packages: WHAT_CHECK [190] and PROCHECK [191].

5.4.3 Colchicine and Other Anti-mitotic Agents

During mitosis, microtubules construct the mitotic spindle, which is responsible for the segregation of aligned chromosomes prior to cell division. As such, microtubules have become the target for a large number of antimitotic agents including antitumor drugs such as the vinca alkaloids, taxanes, epothilones, and colchicine. The method of action of these drugs is to promote or inhibit microtubule polymerization by binding at specific sites on the interface of α/β-tubulin heterodimers. For example, vinblastine, a vinca alkaloid shown in Figure 5.4, binds at the inter-tubulin dimer interface, ultimately resulting in microtubule depolymerization [192]. The binding of the taxanes, on the other hand, results in an overall increase in the spindle microtubule mass with a concurrent reduction in microtubule dynamics [193, 187]. Figure 5.5 shows the binding site of the taxane paclitaxel.

In this work, the focus is on the colchicine molecule. Colchicine is an antimitotic agent that has been considered for chemotherapy applications but is still in pre-clinical cancer research due to its toxicity. It suppresses cell division by inhibiting division of the cell's nucleus. Colchicine binds to the tubulin heterodimer and prevents tubulin from polymerization to form microtubules.

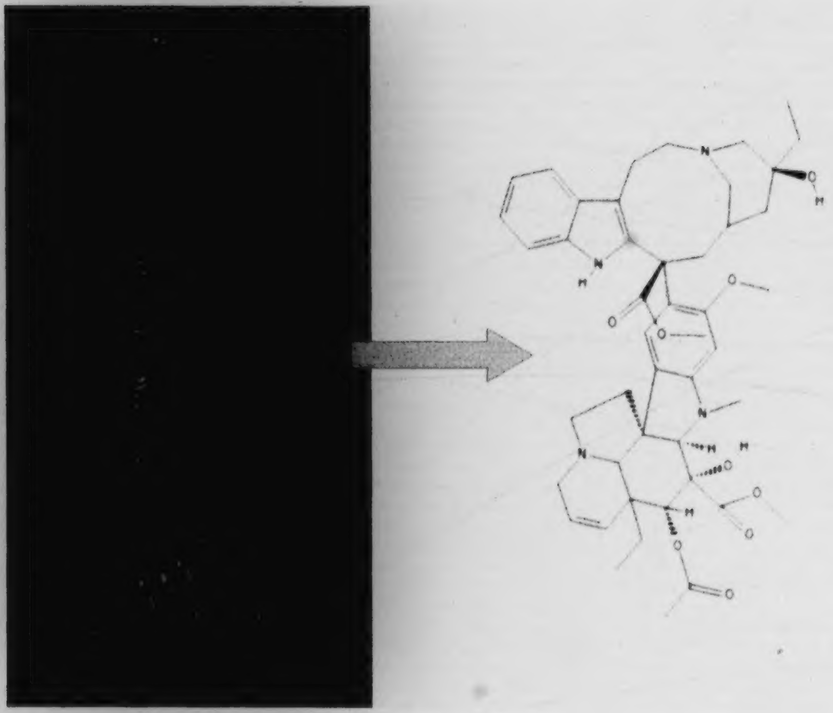


Figure 5.4: Vinblastine binding site located at the inter-tubulin dimer interface.

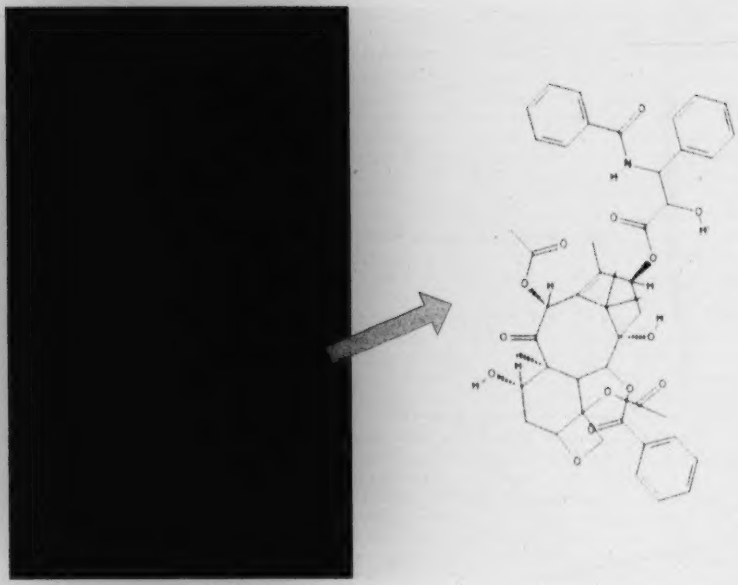


Figure 5.5: α/β-tubulin heterodimers showing the taxane binding site.

5.4.4 The Colchicine Binding Site

Colchicine is a water-soluble alkaloid that, like paclitaxel, binds to α/β-tubulin dimers and blocks cell division thereby inhibiting mitosis. Unlike paclitaxel, the binding of colchicine does not result in microtubule stabilization, but results in their destabilization. Initial structures of a two-heterodimer protofilament, complexed with the stathmin-like domain of RB3 were initially determined (PDB identifier 1FFX) [1]. Recently, this work was followed by an additional structure that identified colchicine as binding between the α- and β-tubulin molecules within the heterodimer itself (PDB identifier 1SA0) [194]. While the binding site for colchicine was shown to be at the interface between α- and β-tubulin, the majority of the interaction is actually restricted to the β-tubulin monomer. Ravelli et al. [194] identified several principal interactions between the bound colchicine and β-tubulin. First were interactions with strands S8 and S9, helix H7 and H8 and loop T7 (see Ref. [185]). These interactions occur as a result of the colchicine binding site being buried within the intermediate domain of β-tubulin. The authors also suggest that interactions between colchicine and α-tubulin's T5 loop could result in the observed stabilization and bending of the α/β heterodimer [195]. They also point out that upon binding of colchicine, the displacements of residues Asp249 and Lys349 are thought to interfere with filament formation. Additional biochemical data have also suggested that Val316 and Cys239 are also involved in colchicine binding [196]. Observations have also been made that suggest that the chemical sensitivity of Cys239 and Cys354, within β-tubulin, is altered upon binding of colchicine[197].

A total of 29 residues of β-tubulin have been identified that are within the 6 Å cut-off range from the bound colchicine, namely: 235-240, 246-257, 312-316 and 347-352. Of these 29 residues, seven positions show differences between the β-tubulin isoforms. All of the observed substitutions occur within the H7 and H8 helix, as well as the S8 and S9 sheets. These positions were initially identified by Ravelli [194] as those that are displaced upon colchicine binding to β-tubulin. The differences here occur over a wide range of β-isoforms, encompassing βIII, βV, βVI, βVII and βVIII (see Table 5.5 and Figure 5.6). Many of these substitutions are conservative, with the exception of the βVII position Val255-Met and the βIII, βVII and βVII position Ala316-

Thr/Cys. Interestingly, there are four positions within the tubulin sequence alignment that have no clear consensus residue over all the β-isoforms, one of which is position 316, which falls within the colchicine binding site and can accommodate either Val or Ile.

Table 5.5: Tubulin isoform interactions with colchicine.

Isoform	Position					
βIII		Cys239-Ser			Ala315-Thr	Thr351-Val
βV		Cys239-Ser			Ala315-Thr	Thr351-Val
βVI	Val236-Ile	Cys239-Ser			Ala315-Thr	Thr351-Val
βVII			Val255-Met	Val313-Ala		
βVIII				Val313-Ala		

Furthermore, when comparing the results of this modeling of the colchicine binding site to the data obtained in the 1SA0 structure [198], no obvious differences can be seen with the exception of Cys239-Ser in βIII, βV and βVI. This was identified by Chaudhuri [197] as being involved with colchicine binding through cysteine labeling studies. This position, while spatially conserved, produces an altered chemical environment, the difference being the presence of a hydroxyl or sulfhydryl moiety. A change like this could be exploited by producing covalently bound forms of a drug, under certain conditions in the form of either a disulfide or ester linkage. Additionally, the substitution Val255-Met within the βVII isoform might alter the positional dynamics of helix H8 and therefore influence colchicine binding in this way. Residues that are present within the interface between H7/H8 and S8/S9, prior to their displacement upon colchicine binding, may also produce interactions that impart varied stability to this region.

Based upon this initial analysis of the ten β-tubulin isoforms, it is clear that the greatest variability occurs within the sequences of βIII, βV, βVI, βVII and βVIII. Interestingly, no variation can be seen, within the drug binding site, in βI, βII or βIV. This may be due to conservation of the binding pocket structure, as a result of the importance of the underlying structure and dynamics of the microtubule. In addition to isoform specificity and drug binding analysis, recent experimental evidence has also suggested that micro-

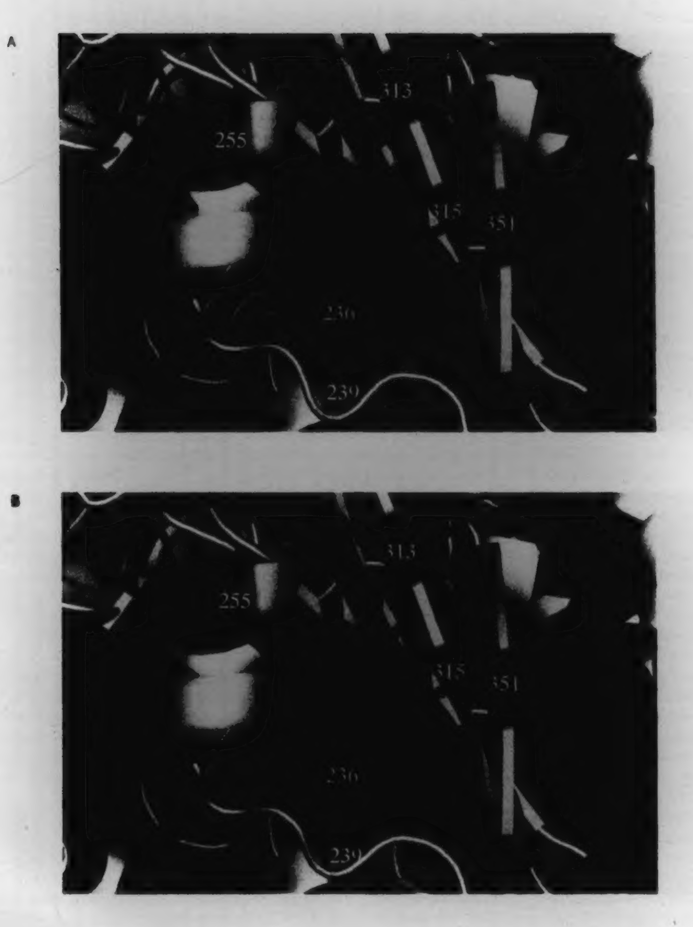


Figure 5.6: Schematic diagram of isoform differences in the colchicine binding site. Models of the βIII, βV, βVI, βVII and βVIII isoforms were superimposed, along with the β chain B from 1SA0[1], and residues containing any atom within 6 Å of the bound colchicine molecule were selected. The amino side chain differences between each of the isoforms within this cut-off are shown as red sticks, while colchicine is shown in green sticks. (A) shows those residues, as they would be observed within the β tubulin consensus sequence. (B) is an identical view as (A), with the exception that each position carries the appropriate amino acid substitution as seen in the isoforms.

tubule dynamics can be changed by the inclusion of tubulin isoforms, in the absence of a bound drug molecule [165]. These results imply that when the equilibrium of microtubule assembly and disassembly is altered, the cell can respond by producing a specific tubulin isoform that will bring the system back to a normal balance. Based on this hypothesis, it can also be proposed that the expression of tubulin isoforms that ultimately leads to microtubule destabilization may also play a role in the ability of cancerous cells to undergo rapid rounds of subsequent cell division.

5.5 Conclusion

Cancer is the result of uncontrolled cell division that can metastasize and cause secondary tumors. Treatment of cancer includes surgery, radiotherapy, and chemotherapy. Chemotherapy uses cytotoxic drugs to treat cancer by stopping the cancer cells from dividing and reproducing uncontrollably. Understanding how to target cancer cell requires understanding the normal cell, its components and processes involved in its growth and reproduction.

This chapter discusses how anti-mitotic agents affects the cell. The resulting cellular phenotype of anti-mitotic drugs is the induction of mitotic arrest leading to apoptosis, making them effective chemotherapeutic agents for targeting rapidly dividing cells. Nevertheless, even the most successful chemotherapy drugs have undesirable side effects that limit their utility. Their drawback is that when these drugs are given systemically, they bind tubulin indiscriminately, leading to the destruction of both cancerous and healthy cells, the consequence of which is the presence of serious side effects in all known cancer chemotherapy applications.

Chapter 6

Free Energy Calculations: Application to the Relative Binding Free Energy of Colchicine and Its Derivatives with the α/β-tubulin Isoforms

In the previous chapters, the role that microtubules play in the cell and in the cell cycle has been discussed. The anti-mitotic agents affecting microtubules, particularly colchicine, have been mentioned. In the following discussion, computational modeling has been employed in order to find derivatives of colchicine as potential anti-cancer drug candidates which would fit in the colchicine binding site of the α/β-tubulin heterodimer. More specifically, a series of free energy perturbation calculations have been performed on the colchicine bound to several α/β-tubulin isoforms as well as on a number of derivatives of colchicine also bound to the same set of α/β-tubulin isoforms to elucidate which of these molecules may be considered to be potential substitute(s) for colchicine as an anti-cancer drug.

6.1 Relative binding energy calculations

6.1.1 Setting up the initial structures

In forming the initial model for the α/β-tubulin heterodimer, only one type of α-tubulin subunit from a soluble dimer was used throughout the simulations and was taken from the RCSB Protein Data Bank with the pdb identifier 1SA0[198]. The ten human β-tubulin isoforms used in the calculations were obtained from Huzil *et al*[164, 199]. The α- and β-monomers were then joined together to form the ten α/β-tubulin dimers. The coordinates of all the missing hydrogen atoms in the pdb structures were added using the DYNAMO[40] program. The geometry of each system was optimized by energy minimization to refine the structure as well as to relieve any bad contacts among atoms due to the creation of hydrogen atom coordinates.

The structure of colchicine is given in Figure 6.1. The two structural analogues of colchicine and the side groups used in this study are given in Figures 6.2 and 6.3, respectively. The modification to the first analogue is done by replacing the $-OCH_3$ in the $C13$ position of the $A-$ring of colchicine by different $-OX$ groups. In the second analogue, the $-OCH_3$ in the $C11$ position of the $A-$ring of colchicine is replaced by different $-OY$ groups and the $-OCH_3$ in the $C-$ring is replaced by a $-SCH_3$. The choice of the side groups is dictated by the availability of the actual colchicine derivative compounds that can be tested against cell lines. The molecular models for all the resulting derivative structures including colchicine were constructed using MOLDEN program[200]. The structures for each of these molecules were then optimized in the gas phase with the GAMESS-US[37] program using the AM1 semi-empirical method[201, 202].

The geometry-optimized colchicine derivatives were positioned and oriented similarly to that of colchicine at the colchicine binding site between the α- and β-tubulin monomers. This procedure produced a total combination of 10 α/β-tubulin dimers × $n-$colchicine/derivatives systems. A suitable number of Na^+ and Cl^- counterions was added near the charged amino acids on the surface of each colchicine/derivative-α/β-tubulin system to neutralize the charges on the tubulin dimer. Each system was then solvated by superimposing a rectangular box of water molecules with a dimension of 115×75×75

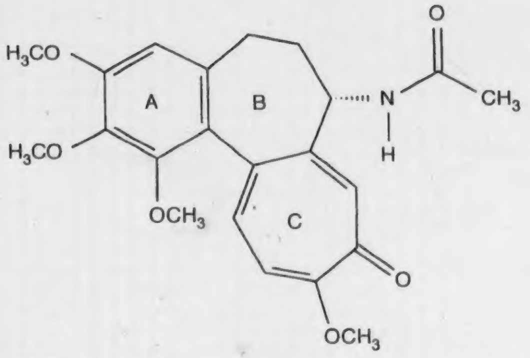
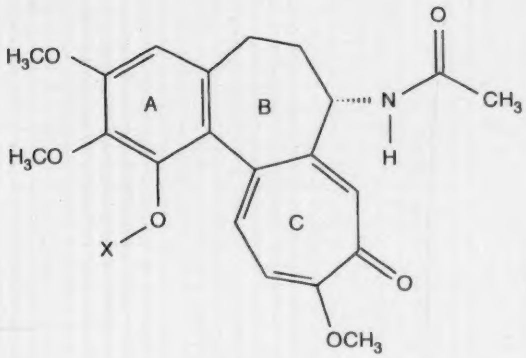


Figure 6.1: The structure of colchicine.

Å that was thermalized at 300K. All water molecules overlapping with the atoms of the colchicine/derivative-α/β-tubulin system, and the counterions, with a buffer distance of 2.5 Å, were removed. Then, a geometry optimization was performed on each system using the conjugate-gradient method in DYNAMO to an rms gradient of 0.1 kJ/Å. Next, all water molecules beyond ~7 Å from the surface of the α/β-tubulin complex were also deleted. This left about 30,000 atoms that included the α/β-tubulin, the colchicine/derivative, the counterions, and approximately two solvation layers of water surrounding the α/β-tubulin complex. A final geometry optimization was performed on each system. All the residues within 18 Å of the colchicine/derivatives were then taken as the dynamically active part of the system while the rest were frozen. The division of the entire system into an active and a frozen part was used in all subsequent simulations.

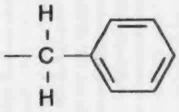
Each of the individual colchicine and derivative molecules was also solvated in a manner similar to the procedure described above. The same rectangular box of water molecules was superimposed to each colchicine and derivative molecules. All water molecules overlapping with the atoms of the colchicine/derivatives and those beyond 30 Å, were removed. The geometry of each system was optimized and partitioned into active and frozen parts similar to the colchicine/derivative-α/β-tubulin system described above.

It should be noted that in all the geometry optimization steps performed, a hybrid quantum mechanical/molecular mechanical (QM/MM) approach was used. The QM/MM method was reviewed by Giao [88]; more recent develop-

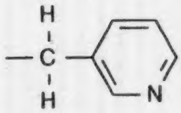

H₃CO
H₃CO
A
B
C
X
O
N
H
O
CH₃
O
OCH₃

1 $-H$
2 $-COCH_3$
3 $-COCH_2CH_3$
4 $-COCH(CH_3)_2$
5 $-CH_2CH_3$
6 $-CH_2CH_2CH_3$
7 $-CH(CH_3)_2$
8 $-CH_2CH(CH_3)_2$
9 $-CH_2CH = CH_2$
10 $-CH_2CH_2OCH_3$

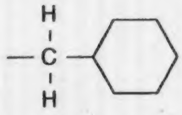
11



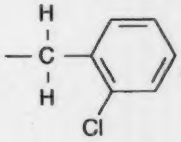
12



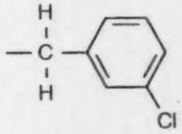
13



14



15



16

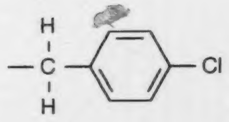
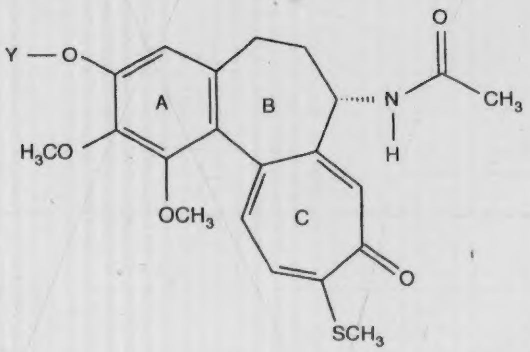


Figure 6.2: The structure of the colchicine analogue and the corresponding X-side groups.



17	$-H$	**20**	$-CH_2CH_2CH_3$
18	$-CH_2CH = CH_2$	**21**	$-\alpha - D - glucose$
1[illegible]	$-CH_2CH_3$	**22**	$-\beta - D - glucose$

Figure 6.3: The structure of the colchicine analogue and the corresponding Y-side groups.

ment were reported by Woodcock et al.[203]. The colchicine and its derivatives were treated as QM atoms embedded within the MM atoms of the α/β-tubulin and water molecules. The AM1 semi-empirical QM potential was used for the atoms of colchicine and derivatives while the OPLA-AA[204] force field was used as the potential for the tubulin atoms and TIP3P[205] for the atoms of water. In this QM/MM approach the active part interacted with the frozen part of the system.

6.1.2 Free energy of binding calculations and their results

The thermodynamic cycle perturbation approach[206] shown in Figure 6.4 shows the scheme that is used as the basis for the determination of the free energies of binding. In this scheme, C and C' represent colchicine and different modifications of colchicine, respectively, ABT represents the α/β-tubulin dimer, and the ΔGs represent free energy changes for the indicated processes. The relative binding of C and C' to ABT is determined by $\Delta\Delta G_{bind} = \Delta G_{C'} - \Delta G_C$. However, in practice, calculation of ΔG_C and $\Delta G_{C'}$ can be very difficult and challenging because the physical binding processes may in-

$$\begin{tikzcd}
C + (ABT) \arrow[r, "\Delta G_C"] \arrow[d, "\Delta G_1"] & C(ABT) \arrow[d, "\Delta G_2"] \\
C' + (ABT) \arrow[r, "\Delta G_{C'}"] & C'(ABT)
\end{tikzcd}$$

Figure 6.4: The thermodynamic cycle.

volve conformational changes or even desolvation steps that are very slow. To overcome these difficulties, ΔG_1 and ΔG_2, although not physically measurable, are calculated instead via free energy perturbation using molecular dynamics. This is accomplished by performing simulations at different values of the perturbation parameter, λ. For example, the free energy difference between two states λ and $\lambda + \delta\lambda$ is calculated by using

$$\Delta G = G_{\lambda+\delta\lambda} - G_\lambda = -RTln \left\langle exp \left(-\frac{(V_{\lambda+\delta\lambda} - V_\lambda)}{RT} \right) \right\rangle_\lambda \tag{6.1}$$

where $G_{\lambda+\delta\lambda}$ and G_λ are the respective free energies of state λ and $\lambda + \delta\lambda$, V_λ and $V_{\lambda+\delta\lambda}$ are the potential energies both evaluated for the same configuration of the system, R is the gas constant and T is the absolute temperature. The pointed brackets denote the average value of the enclosed quantity for each configuration in the trajectory. The total free energy difference is calculated from the sum of the free energy differences of the related intermediate λ states:

$$\Delta G = \sum_{\lambda=0}^{\lambda=1} (G_{\lambda+\delta\lambda} - G_\lambda) \tag{6.2}$$

By using the thermodynamic cycle in Figure 6.4 and the fact that free energy is a state function, the relation, $\Delta\Delta G_{bind} = \Delta G_2 - \Delta G_1$, is justified.

In order to calculate the $\Delta\Delta G_{bind}$, two types of simulations were carried out – one for C and each C', and another for the $C(ABT)$ and each $C'(ABT)$ non-covalent complex. In all the model structures, the initial coordinates were taken from the final step of the structure preparation described above. In all the simulations, prior to the start of free energy perturbation calculations, each system was energy minimized using a conjugate-gradient method to an rms gradient of 0.1 kJ/Å, and then equilibrated for 10 ps at 300 K with molecular dynamics. In order to control the temperature and approximate a canonical ensemble, all molecular dynamics simulations were done with a Langevin algorithm using a friction coefficient of 10^{-1} ps for each atom and a time step of 1 fs. The QM/MM and MM/MM nonbonding interactions were calculated using an atom-based force-switching truncation function with inner and outer cutoffs of 9.0 and 13.0 Å, respectively, and a non-bonded cutoff of 15.0 Å. The QM charges of colchicine and its derivatives in the last step of the equilibration were then saved and used in the subsequent classical free energy perturbation calculations. Since there are interactions among QM and MM atoms, these calculated QM charges have been chosen to be used for the colchicine/derivatives in the presence of MM charges, as opposed to just QM charges calculated in the gas phase (i.e. the influence of MM charges in calculating QM charges is taken into account). These calculated QM charges are believed to be a better representation of actual charges that can be used in the classical free energy perturbation calculations described next. Other schemes for deriving partial atomic charges such as CHELP, CHELPG, Merz-Kollman, and RESP, can also be used [207].

For the classical free energy perturbation calculations, the electrostatic free energy was evaluated separately from the van der Waals energy. For a smooth evaluation of the van der Waals free energies, only the van der Waals well depth was explicitly changed in each step of the perturbation calculations. At the beginning of each simulation, the coordinates from the last step of the equilibration described above were used. The perturbation was immediately performed with varying time lengths of 1.0, 1.5, 2.0 and 2.5 ps for each of the equilibration and data collection steps, using a 1 fs time step. Both the forward ($\lambda = 1 \rightarrow 0$) and backward ($\lambda = 0 \rightarrow 1$) changes in the free energies were evaluated. At each window of equilibration and data collection, λ was varied by 0.05. The average binding free energies and the standard deviations were then calculated from these four free energy perturbation calculations. All molecular dynamics and free energy perturbation calculations are done using DYNAMO.

Tables 6.1 and 6.2 show the calculated changes in the binding free energies with respect to colchicine which was used as a reference point for all the molecules shown in Figures 6.2 and 6.3, respectively. Figures 6.5 and 6.6 give a visual representation of the ranking of the analogues when used as a replacement for colchicine for the ten α/β-tubulin isoforms that were used in the study. The $\Delta\Delta G_{bind}$ with negative values correspond to molecules that are more strongly bound to the α/β-tubulin dimers than colchicine, while those with positive values correspond to those that are less strongly bound to them. Reported large values of $\Delta\Delta G_{bind}$ may be attributed to originate from areas of the protein away from the binding site via relaxation of the entire tubulin structure.

The results of the simulations indicate that there are compounds that potentially could be superior substitutes for colchicine. In Figure 6.5, compound **10** exhibits the best among the compounds studied as it shows that it is more strongly bound to all isoforms with respect to colchicine except with its binding with α/βVIII. In Figure 6.6, compound **19** shows the best binding to seven out of ten α/β-tubulin isoforms.

Table 6.1: Calculated average $\Delta\Delta G_{bind}$ and standard deviations (in kcal/mol) for colchicine derivatives **1-16** against colchicine in 10 α/β-tubulin dimers.

Derivative	I	IIA	IIB	III	IVA
1	-13.2 ± 1.8	5.9 ± 1.5	-2.8 ± 1.6	1.1 ± 1.7	5.5 ± 2.6
2	-9.6 ± 3.3	-24.8 ± 1.6	-7.9 ± 2.2	1.4 ± 2.2	15.1 ± 4.0
3	-19.9 ± 6.2	-10.7 ± 3.9	-17.0 ± 3.9	6.6 ± 2.4	-9.5 ± 4.7
4	-30.0 ± 4.8	-8.3 ± 2.1	-7.0 ± 3.9	2.2 ± 4.0	-8.2 ± 4.2
5	-13.8 ± 3.4	-11.5 ± 2.0	3.5 ± 2.4	5.0 ± 1.7	-7.0 ± 3.5
6	8.6 ± 4.0	4.8 ± 2.1	20.0 ± 1.7	-4.3 ± 1.3	8.1 ± 2.6
7	-2.8 ± 0.8	-9.8 ± 1.7	2.6 ± 1.7	1.2 ± 1.6	1.8 ± 2.3
8	-2.5 ± 2.7	-7.8 ± 1.6	1.1 ± 4.5	2.2 ± 3.0	3.5 ± 3.1
9	2.9 ± 5.8	-7.0 ± 1.9	0.1 ± 3.3	0.3 ± 4.4	14.1 ± 3.7
10	-10.6 ± 4.6	-13.2 ± 3.0	-4.9 ± 4.3	-21.7 ± 1.8	-15.9 ± 4.1
11	37.6 ± 1.6	-0.6 ± 0.8	10.7 ± 1.7	13.1 ± 0.3	3.2 ± 3.4
12	-15.6 ± 1.6	-23.2 ± 2.1	-4.1 ± 1.9	-15.1 ± 1.6	-2.8 ± 2.7
13	0.1 ± 2.3	-5.2 ± 3.4	9.7 ± 1.5	2.3 ± 2.2	-3.7 ± 2.3
14	-0.1 ± 4.0	-11.0 ± 1.9	0.6 ± 1.5	26.3 ± 1.9	26.5 ± 2.7
15	12.4 ± 3.3	-3.5 ± 1.6	-1.4 ± 0.8	6.3 ± 1.6	-4.3 ± 2.3
16	-2.8 ± 4.2	-3.9 ± 0.9	5.5 ± 1.8	-9.1 ± 3.8	-9.9 ± 3.5

Derivative	IVB	V	VI	VII	VIII
1	-16.9 ± 2.7	-19.2 ± 1.0	-10.4 ± 2.9	2.1 ± 1.1	41.4 ± 3.3
2	-5.4 ± 3.9	-13.7 ± 2.9	-10.0 ± 2.2	7.5 ± 1.8	7.7 ± 1.7
3	-20.7 ± 7.6	-6.8 ± 3.5	-12.2 ± 6.5	-8.2 ± 3.3	7.7 ± 2.7
4	-10.0 ± 3.4	-8.3 ± 4.1	-27.9 ± 4.9	-6.7 ± 3.3	0.6 ± 2.4
5	-13.9 ± 2.1	-4.4 ± 3.3	-14.5 ± 3.4	6.7 ± 1.4	22.3 ± 2.2
6	-12.7 ± 5.7	10.8 ± 3.2	-3.9 ± 5.1	5.2 ± 2.3	21.2 ± 1.2
7	-10.2 ± 1.9	-12.7 ± 4.4	-2.2 ± 2.9	9.0 ± 1.3	16.0 ± 0.9
8	-16.5 ± 6.2	2.3 ± 2.5	-29.2 ± 5.4	-0.5 ± 3.8	12.6 ± 1.8
9	4.7 ± 6.3	-5.3 ± 4.1	-4.2 ± 6.0	16.1 ± 2.5	8.0 ± 2.2
10	-15.8 ± 6.1	-18.3 ± 4.2	-29.0 ± 6.5	-3.8 ± 3.7	6.8 ± 3.1
11	-19.1 ± 1.6	-4.2 ± 1.7	7.8 ± 4.2	10.6 ± 0.2	8.4 ± 1.4
12	-30.4 ± 3.3	-10.4 ± 4.8	-20.2 ± 3.5	30.3 ± 2.2	6.2 ± 0.8
13	-18.1 ± 3.6	1.1 ± 4.7	-11.3 ± 6.6	9.3 ± 4.7	12.8 ± 2.5
14	-20.0 ± 6.1	-2.0 ± 3.0	-16.2 ± 3.9	14.6 ± 2.8	37.9 ± 4.4
15	-4.4 ± 3.2	6.3 ± 2.0	-10.4 ± 3.2	7.4 ± 2.2	40.5 ± 1.1
16	-15.5 ± 2.2	-19.8 ± 2.3	-15.4 ± 2.6	8.7 ± 2.2	5.0 ± 0.9

Table 6.2: Calculated average $\Delta\Delta G_{bind}$ and standard deviations (in kcal/mol) for colchicine derivatives **17-22** against colchicine in 10 α/β-tubulin dimers.

Derivative	I	IIA	IIB	III	IVA	IVB	V	VI	VII	VIII
17	7.8 ± 5.1	-1.3 ± 1.4	3.1 ± 4.3	1.9 ± 1.5	26.1 ± 3.6	-6.6 ± 3.8	7.7 ± 2.0	-18.5 ± 5.9	23.9 ± 2.5	7.7 ± 2.9
18	-2.1 ± 2.9	-1.1 ± 3.4	3.3 ± 2.8	16.8 ± 1.4	-5.6 ± 1.9	-10.1 ± 2.3	-6.6 ± 3.1	-15.9 ± 5.1	-0.6 ± 1.3	9.9 ± 2.1
19	-10.3 ± 3.8	3.4 ± 3.2	-11.0 ± 2.3	-0.2 ± 1.5	-15.1 ± 3.2	-6.1 ± 5.1	-12.9 ± 4.2	-33.6 ± 3.4	4.4 ± 2.4	6.6 ± 1.4
20	-1.7 ± 5.1	0.9 ± 1.7	10.6 ± 3.5	7.8 ± 1.8	-6.4 ± 3.9	-8.4 ± 3.7	4.8 ± 2.9	-16.2 ± 4.9	11.0 ± 3.2	11.3 ± 1.4
21	19.3 ± 4.6	25.2 ± 3.4	32.8 ± 2.9	38.0 ± 1.8	32.1 ± 2.4	27.3 ± 3.8	21.3 ± 3.0	11.8 ± 5.7	48.8 ± 4.1	50.9 ± 2.5
22	-4.7 ± 4.8	-2.6 ± 2.5	-0.4 ± 4.4	23.8 ± 2.6	-1.9 ± 4.2	2.2 ± 3.2	15.8 ± 2.7	-9.2 ± 7.0	21.2 ± 5.0	8.1 ± 2.8

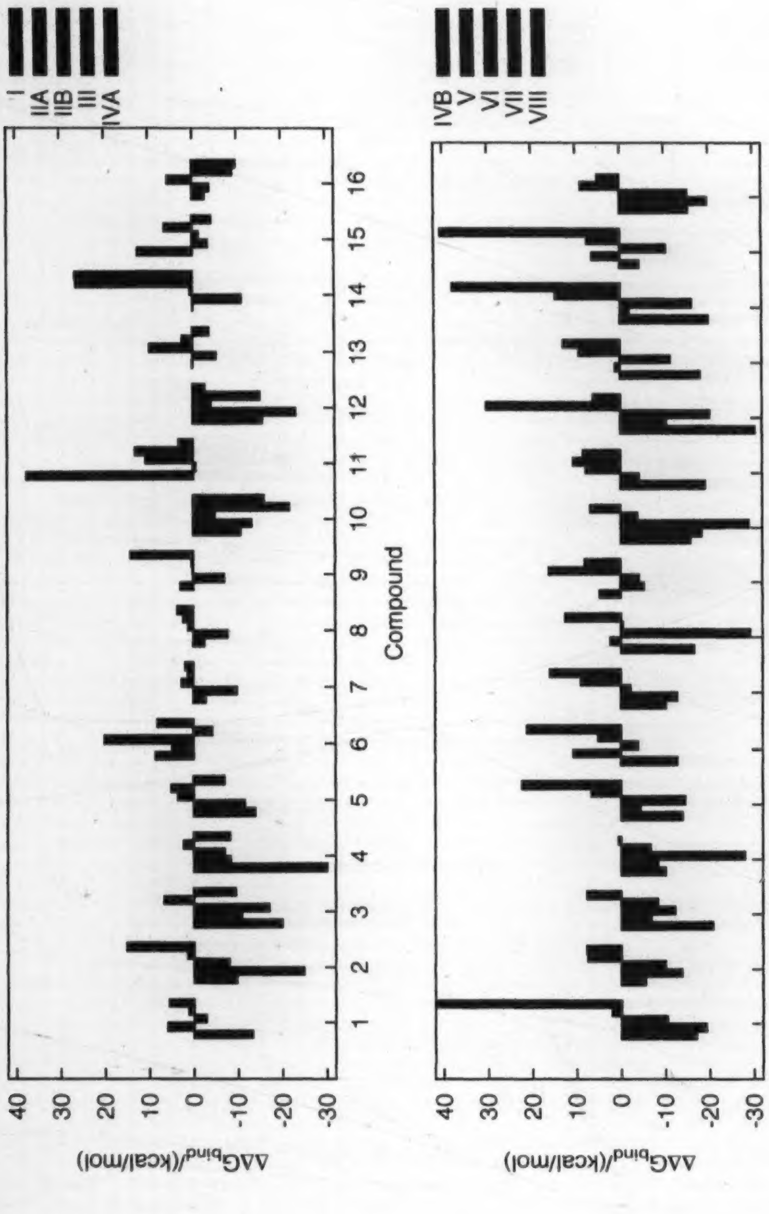


Figure 6.5: α/β-tubulin isoforms and compounds 1-16.

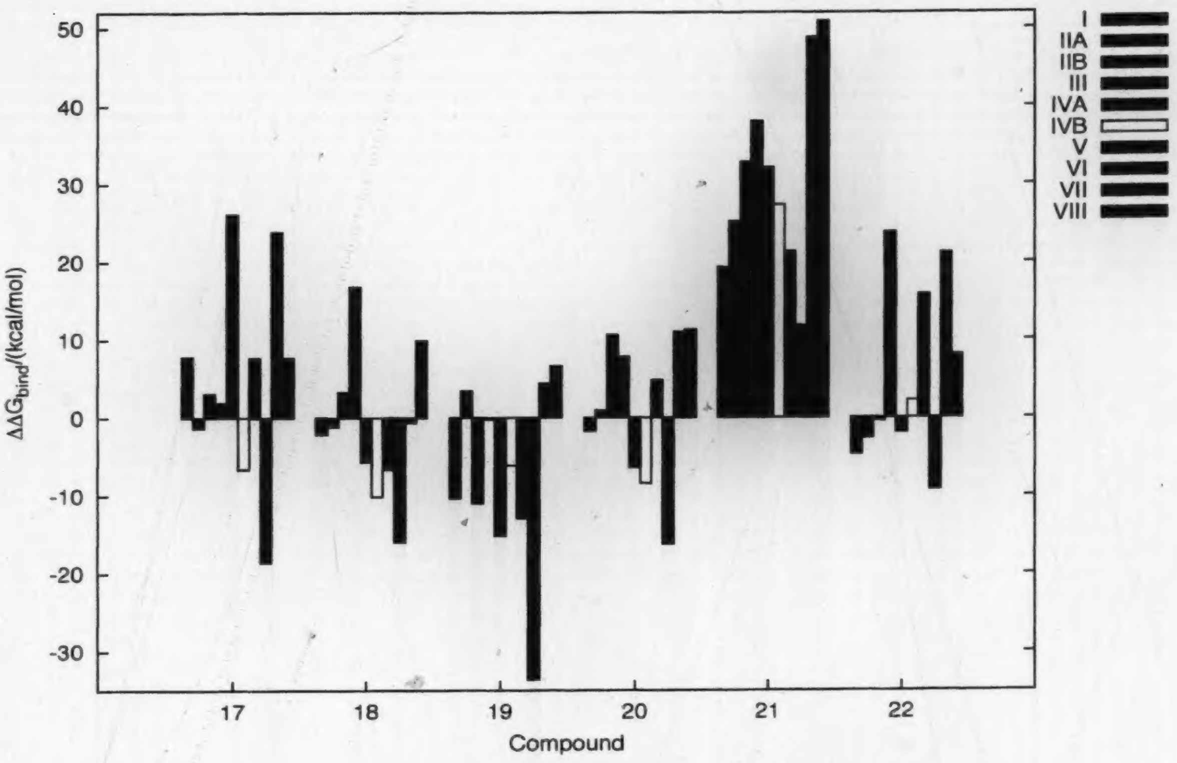


Figure 6.6: α/β-tubulin isoforms and compounds **17-22**.

As can be seen from Figures 6.5 and 6.6, the relative binding energy with respect to colchicine is selective. None of the derivatives exhibits the best or worst binding overall, with the sole exception of compound **21**, that does so in all the 10 α/β-tubulin dimers. This variability is expected and can be attributed to the differences in the amino acid side chains within the colchicine binding site of the 10 α/β-tubulin dimers.

6.2 Conclusion

The estimated values of $\Delta\Delta G_{bind}$ presented in this chapter can serve as a first of the many steps in designing, testing, and identifying specific chemotherapeutic compounds for targeting specific tubulin isoform that are overexpressed in cancer cells. The ultimate goal of cancer research is to develop a drug or treatment regimen that will target only cancer cells and will target them absolutely. The significance of microtubules as a target for chemotherapeutic treatments is outlined in a recent review by Jordan and Wilson [208]. They emphasize the importance of understanding the underlying mechanistic processes of these drugs when they bind to the target protein. While it is clear that a substantial amount of experimental work has to be done on obtaining kinetic data for drug binding to each β-tubulin isoform, the presence of minor variations within the structure of β-tubulin isoforms may provide us with an initial starting point for the development of novel drugs, or the derivatization of existing drugs that have increased specificity. This chapter provided an attempt in this direction by investigating a diversity within a specific group of colchicine derivatives. The results are encouraging and shows a degree of specificity for each tubulin isoform exhibited by the panel of the designed colchicine derivatives. It is expected that these results will be supported by *in vitro* experiments. This type of approach, when brought to a successful completion, may eventually allow us to develop secondary treatments for cancer cell lines that have developed drug resistance, due to mutations or altered expression levels, as a result of standard chemotherapy treatments.

Various mutations exist that affect the amino acid sequence of β-tubulin. Many of these mutations are somatic and develop within the tumour site. A number of common mutations have been identified to occur in the colchicine binding site and they may lead to the development of drug resistence over the course of chemotherapy. To overcome this complication, there are intentions to calculate the binding affinities of these tubulin mutants for the colchicine derivatives discussed in this chapter and select the optimized structures for the particular over-expressed mutant which could eventually result in better cure outcomes.

Chapter 7

DYNAMO and Umbrella Sampling: An Application to Diaminopimelate (DAP)

Peptidoglycan is one of the key component of a bacterial cell wall. It is responsible for giving the cell wall its strength and rigidity. The production of bacterial peptidoglycan can be disrupted by antibiotics such as penicillins and glycopeptides. This disruption of the peptidoglycan biosynthetic pathway leads many researchers to design and synthesize inhibitors of peptidoglycan biosynthesis [209].

One of the components of peptidoglycan is L-lysine. Finding inhibitors that disrupts the biosynthetic pathway leading to the production of L-lysine in bacteria is crucial in the design of drugs that can serve as antibacterial agents. In bacteria, ($2R$,$6S$)-diaminopimelate (or *meso*-DAP) is an intermediate in the biosynthesis of L-lysine. Diaminopimelate epimerase (E.C. 5.1.1.7) converts ($2S$,$6S$)-DAP (or L,L-DAP) into ($2R$,$6S$)-DAP (or *meso*-DAP) via inversion of the configuration at the α-carbon of the amino acid. Figure 7.1 shows the epimerization reaction of L,L-DAP by DAP epimerase. The DAP epimerase active site, which is responsible for this inversion of configuration, contains two cysteine residues: Cys73 and Cys217. The Cys73S$^-$ is the thiolate form which acts as a base for abstracting the H-atom resulting in the deprotonation of the α-carbon. The result is the formation of a planar carbanionic intermediate. The Cys217–SH is the thiol form which acts as an acid for protonating the

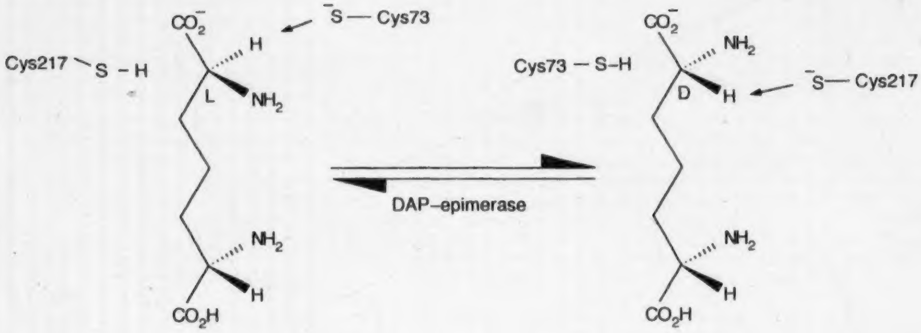


Figure 7.1: The epimerization reaction of L,L-DAP.

intermediate from the other side of the planar α-carbon [210].

Since the DYNAMO [40] program is a collection of library of modules, invoking the correct subroutine or setting up the necessary series of subroutine calls is sometimes challenging. In this chapter, the DYNAMO program is used to calculate a free energy profile. The pieces of the program presented here can serve as template or guide for performing simulations of similar type. In the example that follows, the departure of the α-H from the α-C of the bound L,L-DAP in the L,L-DAP epimerase is used to illustrate the calculation of the free energy profile using umbrella sampling.

7.1 Umbrella sampling to calculate the free energy profile

7.1.1 Setting up the system

The crystallographic structure of the enzyme was taken from the RCSB Protein Data Bank with the pdb identifier 2GKE [211]. This is the crystal structure of DAP epimerase complexed with an irreversible inhibitor L,L-AziDAP. In this structure the catalytic cysteine residues are present as Cys73S$^-$ and Cys217S–H, respectively, in complex with the L,L-AziDAP. The L,L-aziDAP in the crystal structure was replaced with the L,L-DAP, as shown in Figure 7.2. The coordinates of all the missing hydrogen atoms in the pdb structure were

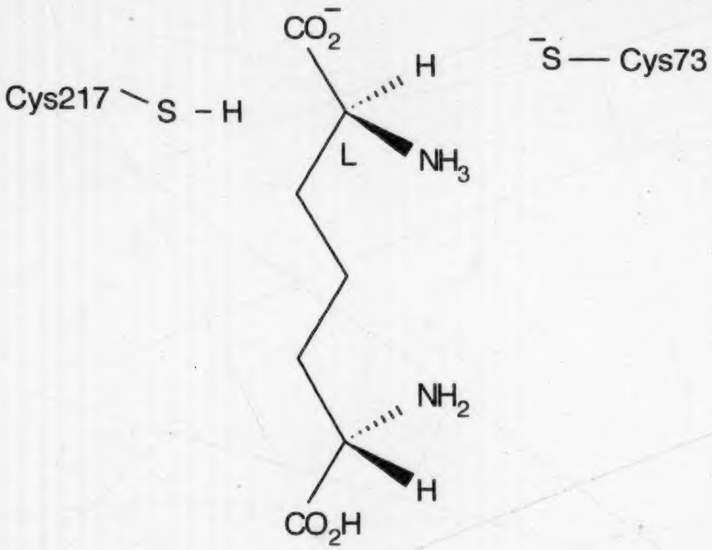


Figure 7.2: The structure of L,L-DAP used in the umbrella sampling in the presence of Cys73–S$^-$ and Cys217–S–H pair.

added using the Dynamo [40] program. The geometry of the system was then optimized by energy minimization to refine the structure and to relieve any bad contacts among the atoms due to the creation of the hydrogen coordinates. A suitable number of counterions was also added to neutralize the charges on the L,L-DAP epimerase.

The refined L,L-DAP epimerase with the L,L-DAP was then solvated by superimposing a cubic box of water molecules with a side of 70 Å that was thermalized at 300 K. All water molecules overlapping with the atoms of the L,L-DAP epimerase system, with a buffer distance of 2.8 Å, including those water molecules beyond 35 Å from the DAP α-C, were removed. This was followed by a geometry optimization using the conjugate gradient method in Dynamo to an rms gradient of 0.1 kJ/Å.

In the calculation of the free energy profile, a hybrid QM/MM approach was used. The L,L-DAP, and residues 73(Cys), 208(Glu), 209(Arg), 217(Cys), 218(Gly), and 219(Ser), were treated as QM atoms and the rest were treated as MM atoms. The AM1 semi-empirical potential was used for the QM atoms while the OPLS-AA [204] force field was used for the MM atoms and TIP3P [205] for the atoms of water.

7.1.2 The reaction coordinate for the free energy profile

To determine the free energy profile for the departure of the α-H from the α-C, the reaction coordinate that was chosen is the bond between the α-H and the α-C of L,L-DAP. This bond is constrained at different values or windows by a constraining potential. Since Cys73–S$^-$ is directly in the vicinity of the leaving α-H, as the bond distance between the α-C and the α-H elongates, it is very likely that the α-H will eventually be captured by the nearby Cys73–S$^-$.

7.1.3 A program illustrating the calculation of the free energy profile

Figures 7.3–7.4 show `PROGRAM PMFWINDOWS`. The program begins by building the MM system via a series of subroutine calls to `MM_FILE_PROCESS()`, `MM_SYSTEM_CONSTRUCT()`, and `MM_SYSTEM_WRITE()`. Then the coordinates L,L-DAP epimerase system are read. The MM energy can be calculated by calling `ENERGY()`. In order to define the atoms to be treated as QM atoms, a call to `MOPAC_SETUP()` is made by passing the necessary arguments. The selected QM atoms are assigned to `SELECTION` by invoking the function `ATOM_SELECTION()` passing to it the `RESIDUE_NUMBER` that corresponds to those defined in the coordinate file `dapE_ions_eq.crd`. In this case, the amino acid residues to be treated as QM are 73, 208, 209, 217, 218, 219, 275, respectively. Optionally, the QM/MM energy can be calculated.

Once the system has been set up, what is next is the umbrella sampling. The number of windows, `NWINDOWS`, represents the total number of simulations to be performed for the umbrella sampling. `NWINDOWS` is also needed to calculate the point in the reaction coordinate where the biasing potential is centered. In this case, the reaction coordinate is related to `DISTANCE`, which is the distance between the α-H and the α-C of DAP. In the example given in

Figure 7.4, there are 21 windows with the `DISTANCE` ranging from 1.30 to 2.30 Å.

The constraint point is defined by invoking `CONSTRAINT_POINT_DEFINE()` and the constraint is defined by calling `CONSTRAINT_DEFINE()`. In this case, atom 4226 is the α-C and atom 4227 is the α-H. The constraint is set as the distance between these atoms with the umbrella potential of 2000 kJ/mol. The file `dapE_rc1_win_##` is where the information for the current window is saved. The data that is stored in this file are those that are collected during the molecular dynamics between the subroutine calls `CONSTRAINT_WRITING_START` and `CONSTRAINT_WRITING_STOP`. Once all the sampling of windows has been done, all the windows are processed by piecing them together using the weighted histogram analysis method (WHAM) in order to calculate the free energy profile. Figure 7.5 shows how to perform a WHAM. Data are read from each `dapE_rc1_win_##` and then passed on to subroutine `WHAM_ANALYZE()`. The temperature is required by `WHAM_ANALYZE()`. It must be the same temperature used during the sampling of each window.

Figure 7.6 shows the energy profile for the departure of α-H from α-C. It can be seen that there is a minimum located at 1.49 Å corresponding to α-H–α-C distance where it is being stabilized by the Cys73–S$^-$, and by the approaching hydrogen of the Cys21S–H. Stretching the α-H–α-C beyond 1.49 Å increases the energy up until 1.59 Å where a transition occurs. Beyond 1.59 Å, a dramatic decrease in energy can be observed. This can be interpreted that α-H of DAP has been completely transfered to the Cys73S$^-$ and that the inversion of configuration has occurred due to the attack of the hydrogen from the Cys21S–H.

7.2 Conclusion

This chapter shows a simple example about how the DYNAMO program can be used for calculating a free energy profile. The decision on how the QM and the MM region is divided is left to the user. The selection of QM atoms (or even MM atoms) can be done by either using their atom numbers or as a group using their residue numbers. Provided that a reaction coordinate is defined, the program that is presented here can be adapted or modified for

```
PROGRAM PMFWINDOWS

USE DYNAMO
IMPLICIT NONE

INTEGER,            PARAMETER :: NWINDOWS  = 21
REAL ( KIND = DP ), PARAMETER :: INCREMENT  =  0.05_DP
REAL ( KIND = DP ), PARAMETER :: MINIMUM    =  1.30_DP

INTEGER               :: I
CHARACTER ( LEN = 2 ) :: ISTRING
REAL ( KIND = DP )    :: DISTANCE

CALL DYNAMO_HEADER
CALL TIME_PRINT

CALL MM_FILE_PROCESS     ( "dapE_ions.opls_bin", "dap-protein.opls" )
CALL MM_SYSTEM_CONSTRUCT ( "dapE_ions.opls_bin", "dapE_ions.seq"    )
CALL MM_SYSTEM_WRITE     ( "dapE_ions.sys_bin" )

CALL COORDINATES_READ    ( "dapE_ions_eq.crd" )

! Calculate the MM energy
CALL ENERGY

! Define the active site to be QM.
CALL MOPAC_SETUP ( "AM1", CHARGE = -1, MULTIPLICITY = 1,  &
                 SELECTION = ATOM_SELECTION(              &
                 RESIDUE_NUMBER = (/ 73,208,209,217,218,219,275 /), &
                           PRINT=.TRUE. ) )

CALL ENERGY      ! Calculate QM/MM energy
.
.
.  ! continued in the next Figure.
```

Figure 7.3: Program illustrating the generation of windows for calculating the free energy profile.

```
.
.
.
DO I = 1, NWINDOWS
   CALL RANDOM_INITIALIZE ( 680500 + I  )
   DISTANCE = MINIMUM + INCREMENT * REAL ( I-1, DP )

   CALL ENCODE_INTEGER ( I, ISTRING, "(I2)" )
   CALL CONSTRAINT_INITIALIZE

   ! Define constraints for the reaction coordinate (DAP alp-C...alp-H)
   CALL CONSTRAINT_POINT_DEFINE( ATOM_SELECTION ( ATOM_NUMBER = (/ 4226 /) ) )
   CALL CONSTRAINT_POINT_DEFINE( ATOM_SELECTION ( ATOM_NUMBER = (/ 4227 /) ) )
   CALL CONSTRAINT_DEFINE ( "DISTANCE", 2000.0_DP, DISTANCE,    &
                          FILE =  "dapE_rc1_win_"//TRIM ( ISTRING ) )
   CALL CONSTRAINT_PRINT

   CALL VELOCITY_ASSIGN  ( 300.0_DP, REMOVE_TRANSLATION = .FALSE. )

   CALL CONSTRAINT_WRITING_START
   CALL DYNAMICS_OPTIONS ( 0.001_DP, 50, 50, SAVE_FREQUENCY = 50)
   CALL LANGEVIN_VERLET_DYNAMICS ( 300.0_DP, 20.0_DP )
   CALL CONSTRAINT_WRITING_STOP

   CALL COORDINATES_WRITE ( "dapE_am1.crd_"//TRIM ( ISTRING ) )
END DO

CALL TIME_PRINT
END PROGRAM PMFWINDOWS
```

Figure 7.4: Program illustrating the generation of windows for calculating the free energy profile (contd).

```
PROGRAM DAPE_RC1_WHAM

USE DYNAMO
IMPLICIT NONE

INTEGER,            PARAMETER ::    NBINS = 100
INTEGER,            PARAMETER :: NWINDOWS =  21

REAL ( KIND = DP ), PARAMETER :: T     = 300.0_DP

CHARACTER ( LEN = 3 ) :: ISTRING
INTEGER               :: I, J

CHARACTER ( LEN = 32 ), DIMENSION(1:NWINDOWS) :: DATA

CALL DYNAMO_HEADER
CALL TIME_PRINT

CALL MM_SYSTEM_READ ( "dapE_70A_sph.sys_bin" )

! Set up the file names.
WRITE( OUTPUT, "(/A)" ) "Files read for WHAM : "

J = 130
DO I = 1, NWINDOWS
   CALL ENCODE_INTEGER ( J, ISTRING, "(I3)" )
   WRITE( OUTPUT, "(2X,A)" ) "dapE_rc1_win_"//TRIM ( ISTRING )
   DATA(I) =                 "dapE_rc1_win_"//TRIM ( ISTRING )
   J = J + 5
END DO

! Calculate the PMF.
CALL WHAM_ANALYSE ( DATA, NBINS, T, ITERATIONS = 2000 )

! Finish up.
CALL TIME_PRINT

END PROGRAM DAPE_RC1_WHAM
```

Figure 7.5: A program to perform WHAM.

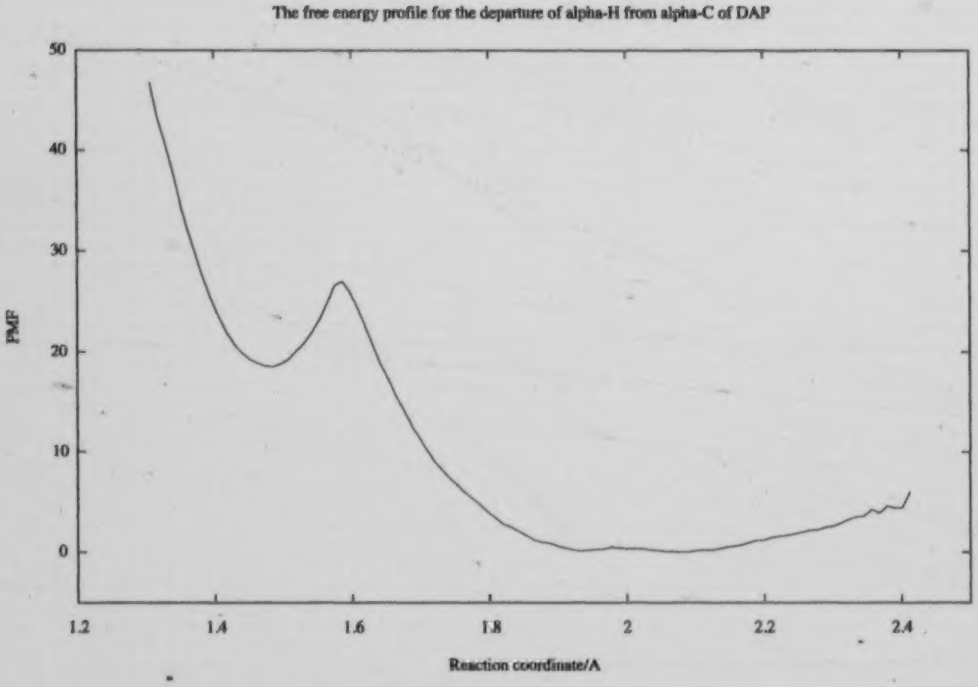


Figure 7.6: The energy profile for L,L-DAP.

application to other systems.

Chapter 8

The DYNAMO Library and the GAMESS-US Program

A hybrid quantum (QM) and molecular mechanical (MM) method allows for the study of a chemical system in which a small part of it can be represented quantum mechanically, and the rest molecular mechanically. The QM region might involve that part of the system in which chemical reactions (that is changes in the distribution of electrons between atoms in the process of bond breaking and bond formation) are taking place. Such a region would be difficult, if not impossible, to be modeled by molecular mechanics.

The DYNAMO library [40] has been developed for the simulation of molecular systems using hybrid QM/MM potentials. However, the DYNAMO library is limited to semi-empirical QM potentials. Even with the recent release of pDYNAMO [212], a descendant DYNAMO library written in Python and C, in addition to semi-empirical QM potentials the library is still limited to density functional theory QC method. In order to use more advanced QM potential, one must either write computer codes for these QM potentials and include it in the library or use already existing codes and interface it to the DYNAMO library. The latter route is chosen in this work. With the availability of computer programs such as GAMESS-US, advanced QM potentials can be exploited and used as alternatives to semi-empirical QM potential.

The DYNAMO library is written in FORTRAN 90 while the GAMESS-US program is written in FORTRAN 77. A good knowledge of FORTRAN 77/90 and programming in general is necessary in order to successfully accomplish interfacing the two programs. A discussion of FORTRAN 77/90 is beyond the scope of this thesis. However, since the DYNAMO library is a collection of modules written in FORTRAN 90, it is worth mentioning the structures of the FORTRAN 90 main program and modules. Whenever necessary, certain elements of FORTRAN 90 that are relevant to the discussion are presented to better explain the segments of computer codes that were developed. For more details on FORTRAN 90, the reader is referred to the book by Metcalf and Reid [213].

8.1 FORTRAN 90

8.1.1 The Main Program Structure

Figure 8.1 shows the general structure of a FORTRAN 90 main program. A FORTRAN 90 program always starts with the keyword `PROGRAM` followed by a program name and ends with the keywords `END PROGRAM` and the program name. Comment lines in FORTRAN 90 are preceded by the `!` symbol. The modules that are needed by the main program, if any, are indicated by the keyword `USE` followed by the module name. The keywords `IMPLICIT NONE` must be present. This is followed by the *data specification* section where constants and variables of the program are declared, then by the *execution section* where calls to subroutines and functions are made. The subroutines and functions are declared in the *subprograms* section.

8.1.2 Module Structure

The structure of a module in FORTRAN 90 is almost identical the structure of the main program, but with the keyword `MODULE` at the start and the keywords `END MODULE` at the end. Figure 8.2 shows the general structure of a FORTRAN 90 module. A module has data specification part and could contain internal subroutines. However, unlike the main program, a module does not have any executable statements between the data specification and the keyword `CONTAINS`. A module can only contain declarations and subroutines to be used by other modules and programs. Consequently, a module cannot exist alone and must be used in a main program or in conjunction with other

```
PROGRAM program-name
! Description of the program

  USE module-name

  IMPLICIT NONE

  [Data Specification]

  [Execution Section]

  [Subprograms]

END PROGRAM program-name
```

Figure 8.1: FORTRAN 90 program structure.

```
MODULE module-name
   USE [other modules]
   IMPLICIT NONE
   [Data Specification]
   CONTAINS
      SUBROUTINE ONE
      ...
      END SUBROUTINE ONE
              .
              .
              .
      SUBROUTINE N
      ...
      END SUBROUTINE N
END MODULE module-name
```

Figure 8.2: FORTRAN 90 module structure.

modules. Modules are compiled separately from the main program unit and usually saved as a separate file. Modules can be archived in a library which in turn can be used by calling it from the main program.

8.2 The DYNAMO Module Library

The DYNAMO library is a collection of FORTRAN 90 modules that can be used for performing QM, MM and hybrid QM/MM simulations. Using the library requires that the main program be written in order to call required modules. The DYNAMO library has modules for performing a wide range of different types of molecular calculation, including the capability to perform calculations of the potential energy of a system using QM, MM, and hybrid QM/MM potentials [40, 214]. It also contains within the modules the utilities to perform molecular dynamic simulations. The DYNAMO library uses semi-empirical MNDO [44], AM1 [201], and PM3 [52, 53] methods for the QM potentials, and the OPLS-AA [85] force field for the MM potentials. The main source of documentation for the modules in the DYNAMO library may be found in the book by Field [214], which describes in details the capabilities of the program.

Only the topics that are relevant to the use of the DYNAMO library are presented for the following discussions. In order to utilize subroutines and functions embedded in the different modules within the DYNAMO library, one must write a FORTRAN 90 program that makes calls to these subroutines and functions. An example of a FORTRAN 90 program making use of the DYNAMO library is shown in Figure 8.3. The first two declaration statements at the start of the program, `USE DYNAMO` and `IMPLICIT NONE`, are necessary. The line `USE DYNAMO` makes the data structures, functions, and subroutines that exists in other FORTRAN 90 modules visible to the main program. This means that all public data structures and procedures of the DYNAMO library are accessible to the program in Figure 8.3, such as calls to `DYNAMO_HEADER`, `TIME_PRINT`, `MM_FILE_PROCESS`, and so on. The line `IMPLICIT NONE` forces all variables to be explicitly typed in the program, such as the declaration of the two-dimensional array `TMPCRD` in Figure 8.3. Calls are made to the different subroutines and functions, if necessary, based on how the user planned the simulation to be performed. The program will terminate once the `END PROGRAM` has been reached.

8.2.1 The Dynamo Coordinate, Sequence, and Force Field Files

Like all other computer programs for molecular simulations, DYNAMO library uses its own set of file handling subroutines. These subroutines process the information contained in the file for the program to function properly. There are files that the user must prepare in order to use the DYNAMO library. These are the *force field file* which contains the OPLS force field to use for the system, the *sequence file* which describes the composition of the molecular system such as the number of subsystems, the number of residues, and the names of each residue as it appears on the *coordinate file*. The *coordinate file* contains the total number of atoms and the their individual coordinates plus the number of residues and subsystem consistent with those declared in the sequence file. Figure 8.4–8.6 show examples of the structure of the force field, sequence and coordinate files for water dimer. These files are used in the main program in Figure 8.3.

As shown in Figure 8.3, in order to use DYNAMO library, calls must be made to subroutines contained in modules. These subroutines may or may not require arguments. A subroutine argument in FORTRAN 90 may be mandatory or optional and declared using the `INTENT` keyword. The `INTENT` can be `INTENT(IN)`, `INTENT(OUT)`, or `INTENT(INOUT)`, and if optional, the keyword `OPTIONAL` is used. For example, a call to `MM_FILE_PROCESS` would require the argument files "`water.opls`" and the optional "`water.opls_bin`" to be passed.

8.3 The GAMESS-US Program Input File

GAMESS-US [37] program allows the use of several advanced quantum mechanical methods for electronic structure calculations. The detailed capabilities of the GAMESS-US program are not discussed here. The relevant information required for the interface with the DYNAMO code is the structure

```
!---------------
PROGRAM EXAMPLE1
!---------------

! . Module declarations.
USE DYNAMO

IMPLICIT NONE

! . Local array declarations.
REAL ( KIND = DP ), ALLOCATABLE, DIMENSION(:,:) :: TMPCRD

CALL DYNAMO_HEADER     ! . Print out the header.
CALL TIME_PRINT        ! . Initialization.

! . Process the MM file for water.
CALL MM_FILE_PROCESS ( "water.opls_bin", "water.opls" )

! . Construct the system file for water.
CALL MM_SYSTEM_CONSTRUCT ( "water.opls_bin", "water_dimer.seq" )

! . Write out the system file.
CALL MM_SYSTEM_WRITE ( "water_dimer.sys_bin" )

! . Read in the coordinates.
CALL COORDINATES_READ ( "water_dimer.crd" )

! . Write out information about the atom data.
CALL ATOMS_FORMULA
CALL ATOMS_SUMMARY

ALLOCATE ( TMPCRD(1:3,1:NATOMS) )      ! . Allocate the temporary array.

! . Assign the values of ATMCRD to TMPCRD
TMPCRD = ATMCRD

DEALLOCATE ( TMPCRD )      ! . Deallocate the temporary coordinate array.

CALL TIME_PRINT       ! . Finish up.

END PROGRAM EXAMPLE1
```

Figure 8.3: An example of user-written FORTRAN 90 program.

```
! . OPLS MM Definition File for Water
MM_Definitions OPLS_AA 1.0

Types    ! . Atom Type Definitions.
! Atom Name     Atomic Number       Sigma       Epsilon
HW                    1           0.00000      0.00000
OW                    8           3.15061      0.15210
End

! . Electrostatics and Lennard-Jones Options.
Electrostatics Scale 0.5
Lennard_Jones  Scale 0.5

Units kcal/mole  ! . Units specification.

Residues         ! . Residue Definitions.
Residue Water
! # Atoms, Bonds and Impropers.
  3  3  0
O        OW      -0.834
H1       HW       0.417
H2       HW       0.417

O H1 ; O H2 ; H1 H2

End

Parameters  ! . Parameter Definitions.

Bonds
! Atoms      FC     Equil.
HO OH      553.0    0.9450
HW OW      529.6    0.9572
HW HW       38.25   1.5139
End

Angles
! Atoms      FC     Equil.
HW OW HW   34.05    104.52
HW HW OW    0.0      37.74
End

End

End
```

Figure 8.4: DYNAMO "`water.opls`" force field file.

```
Sequence
1
Subsystem WATER_DIMER
2
WATER 2
End
End
```

Figure 8.5: DYNAMO "water_dimer.seq" sequence file.

```
!===================================================================
     6     2     1 ! # of atoms, residues and subsystems.
!===================================================================
Subsystem     1  WATER_DIMER
     2 ! # of residues.
!===================================================================
Residue     1  WATER
     3 ! # of atoms.
     1   O        8     1.4190191945    0.0000000000    0.0597044085
     2   H1       1     1.6119040717    0.0000000000   -0.8763193220
     3   H2       1     0.4450228916    0.0000000000    0.0898242033
!-------------------------------------------------------------------
Residue     2  WATER
     3 ! # of atoms.
     4   O        8    -1.3129919772    0.0000000000   -0.0309595744
     5   H1       1    -1.8699678664    0.7570000000    0.1651089311
     6   H2       1    -1.8699678664   -0.7570000000    0.1651089311
!===================================================================
```

Figure 8.6: DYNAMO "water_dimer.crd" coordinate file.

of the GAMESS-US input file. An example of an input file shown in Figure 8.7. The GAMESS-US controls are enclosed between the `$<KEYWORD>` and `$END`. These controls tell the program what kind of calculation to perform. For example, the `$SYSTEM` group specifies the computer related options, the `$CONTRL` group specifies the chemical control data, the `$BASIS` group specifies the basis set to use, `$SCF` controls the HF-SCF wavefunction, and the `$DATA` group contains the title, symmetry, the molecule and its geometry.

```
 $SYSTEM TIMLIM=999999 MWORDS=2 MEMDDI=10 $END
 $CONTRL UNITS=ANGS    INTTYP=HONDO       $END
 $CONTRL SCFTYP=RHF    MAXIT=200          $END
 $CONTRL RUNTYP=GRADIENT                  $END
 $CONTRL ICHARG=0                         $END
 $CONTRL   MULT=1                         $END
 $BASIS  GBASIS=STO NGAUSS=3              $END
 $SCF    DIRSCF=.T.  DIIS=.T. SOSCF=.F.   $END
 $DATA
  TITLE 7 QUANTUM ATOMS
C1
   H      1        1.8499324395      1.7605379372       0.0316343778
   N      7        1.2120499135      2.5133065208      -0.1309858624
   H      1        0.3731365764      2.3739377103       0.3934669916
   CT     6        1.7958893542      3.8572672022      -0.0198728239
   HT1    1        2.2629249755      4.1659689647      -0.9196255789
   HT2    1        0.9281008740      4.5149231432       0.0489092305
   HT3    1        2.4843966832      3.9990044952       0.8155715841
 $END
```

Figure 8.7: An example of GAMESS-US input file.

8.4 The GAMESS-US Modules for DYNAMO

In this work, several modules were written and added to the the DYNAMO library. These modules, listed in Table 8.1, are designed for the exchange of data between DYNAMO and GAMESS-US. It must be noted here that in this version of the DYNAMO and GAMESS-US program, the existing call to the semi-empirical QM in the original DYNAMO version was intentionally removed to reuse existing variables pertaining to the the handling of the quantum atoms and their relationship with the molecular mechanics atoms.

This also allows for minimum modifications of other DYNAMO modules that handle various drivers for simulations, in particular the modules that handle molecular dynamics protocols. In the following discussions, the definition of each of the GAMESS-US modules are presented.

Table 8.1: The GAMESS-US modules added to the DYNAMO module library.

`GAMESS_DATA`	handles data about the QM atoms
`GAMESS_CONTROLS`	handles the input file control groups
`GAMESS_COORDINATES`	handles writing atomic coordinates for GAMESS-US input file
`GAMESS_HAMILTONIAN`	specifies the wavefunction or density functional to use and set up the calculation
`GAMESS_ENERGY`	handles the GAMESS-US QM energy
`GAMESS_GRADIENTS`	handles the GAMESS-US QM gradients

8.4.1 The GAMESS-US Data Module

The definition of the GAMESS-US data module is

```
MODULE GAMESS_DATA

! . Type definitions.
TYPE QMATOM_TYPE
   INTEGER              :: ATOM, BFIRST, BLAST, COPY, NUMBER, PARTNER
   LOGICAL              :: QBOUNDARY
   REAL ( KIND = DP )   :: LENGTH
END TYPE QMATOM_TYPE

! . QM atom data.
TYPE(QMATOM_TYPE), ALLOCATABLE, DIMENSION(:) :: QMATOM

! . Data declaration
CHARACTER ( LEN = 3 )  :: HAMILTONIAN
CHARACTER ( LEN = 32 ) :: BASIS
CHARACTER ( LEN = 6 )  :: DFTFUNC
```

```
INTEGER                 :: MULTIP   = 1, TOTCHG = 0
INTEGER                 :: NBRY
LOGICAL                 :: QUSEBAQM = .FALSE.
LOGICAL                                :: CUTCOVALENTBOND
LOGICAL, ALLOCATABLE, DIMENSION(:) :: ATOMSELECTION

CONTAINS
   SUBROUTINE GAMESS_DATA_INITIALIZE
   END SUBROUTINE GAMESS_DATA_INITIALIZE

   FUNCTION GAMESS_DATA_ATOM_POSITION ( QATOM, COORDINATES )
      REAL ( KIND = DP ), DIMENSION(1:3)  :: GAMESS_DATA_ATOM_POSITION
      INTEGER,                            INTENT(IN) :: QATOM
      REAL ( KIND = DP ), DIMENSION(:,:), INTENT(IN) :: COORDINATES
   END FUNCTION GAMESS_DATA_ATOM_POSITION ( QATOM, COORDINATES )

   FUNCTION GAMESS_DATA_LA_GRADIENT ( QATOM, COORDINATES, GRADIENT )
      REAL ( KIND = DP ), DIMENSION(1:3) :: GAMESS_DATA_LA_GRADIENT
      INTEGER, INTENT(IN)                :: QATOM
      REAL ( KIND = DP ), DIMENSION(1:3), INTENT(IN) :: GRADIENT
      REAL ( KIND = DP ), DIMENSION(:,:), INTENT(IN) :: COORDINATES
   END FUNCTION GAMESS_DATA_LA_GRADIENT ( QATOM, COORDINATES, GRADIENT )

END MODULE GAMESS_DATA
```

This module defines an allocatable one-dimensional array data type `QMATOM` of type `QMATOM_TYPE` which handles data about the QM atoms. The QM method to use (e.g. RHF, MP2, or DFT), which is three characters long, are contained in `HAMILTONIAN`. The `BASIS` contains the basis set to use, up to a maximum length of 32 characters. The way the basis set should be provided follows the format as specified in the `$BASIS` group in the GAMESS-US input file. At the moment, the basis sets that can be used are the GAMESS-US built-in basis sets. For example, to use the `6-311` basis set, the string "`N311 NGAUSS=6`" must be used. By default, the multiplicity `MULTIP` and total charge `TOTCHG` of the system are assigned `1` and `0`, respectively. The variables `MULTIP` and `TOTCHG` correspond to `MULT` and `ICHARG` in the GAMESS-US input file, respectively. The integer `NBRY` is used to hold the number of boundaries that are determined if a QM/MM calculation is employed. The logical variables

QUSEBAQM, **CUTCOVALENTBOND**, and **ATOMSELECTION** are also defined to handle conditions and selections.

The module also contains one subroutine and two functions. The subroutine **GAMESS_DATA_INITIALIZE**, as the name implies, initializes the data structures **ATOMSELECTION** and **QMATOM** of the module. The function **GAMESS_DATA_ATOM_POSITION** takes in a quantum atom index and its coordinates and returns its position in the system. The function **GAMESS_DATA_LA_GRADIENT** handles the coordinates and gradient of the link quantum atom.

8.4.2 The GAMESS-US Controls Module

The definition of the GAMESS-US controls module is

```
MODULE GAMESS_CONTROLS

CONTAINS
   SUBROUTINE GAMESS_CONTROLS_WRITE ( METHOD, BASIS, DFTFUNCTIONAL, &
                                      CHARGE, MULTIPLICITY )
      CHARACTER ( LEN = * ), INTENT(IN)           :: METHOD, BASIS
      CHARACTER ( LEN = * ), INTENT(IN), OPTONAL  :: DFTFUNCTIONAL
      INTEGER,               INTENT(IN), OPTIONAL :: CHARGE, MULTIPLICITY
   END SUBROUTINE GAMESS_CONTROLS_WRITE

END MODULE GAMESS_CONTROLS
```

This module is a utility module that prepares the control groups of the GAMESS-US input file. It contains the subroutine **GAMESS_CONTROLS_WRITE** that requires the arguments **METHOD**, and **BASIS** and the optional arguments **CHARGE** and **MULTIPLICITY**. The optional argument **DFTFUNCTIONAL** is only required if the string that is passed to the **METHOD** argument is "**DFT**".

8.4.3 The GAMESS-US Coordinates

The definition of the GAMESS-US coordinates module is

```
MODULE GAMESS_COORDINATES

CONTAINS
  SUBROUTINE GAMESS_COORDINATES_WRITE ( FILE, DATA, SELECTION, CUTCOVALENT )
    CHARACTER ( LEN = * ), INTENT(IN), OPTIONAL :: FILE
    LOGICAL,                           OPTIONAL :: CUTCOVALENT
    LOGICAL, DIMENSION(1:NATOMS),    INTENT(IN), OPTIONAL :: SELECTION
    REAL ( KIND = DP ), DIMENSION(1:3,1:NATOMS), INTENT(IN), OPTIONAL :: DATA
  END SUBROUTINE GAMESS_COORDINATES_WRITE

END MODULE GAMESS_COORDINATES
```

This module handles the writing of the necessary coordinate files that are used by GAMESS-US. The module contains only the subroutine `GAMESS_COORDINATES_WRITE`. All the arguments of the subroutine are optional. The `FILE` argument holds the name of the file and the `DATA` holds the coordinates of the atoms. The `SELECTION` argument is a logical array that determines whether the atom is QM or MM atom and the `CUTCOVALENT` determines whether a covalent bond between a QM and MM atoms is cut.

The subroutine GAMESS_COORDINATES_WRITE writes two important files, **gamess.inp** and **dynamomm**. The **gamess.inp** is the GAMESS-US input file that is used to calculate the QM potential of the selected part of the system assigned to be in the QM region. The **dynamomm** file contains information about the atoms considered as MM atoms of the system. The **dynamomm** file contains the coordinates and the classical charges of the MM atoms which are then read and incorporated in the GAMESS-US program as perturbations to the QM potentials of the QM atoms.

If a covalent bond exists between the QM and MM atoms of the system (i.e. `CUTCOVALENT=.TRUE.`), then a hydrogen link-atom is added to the **gamess.inp** file to satisfy the valency of the bond that is broken between the QM and MM regions.

8.4.4 The GAMESS-US Hamiltonian Module

The definition of the GAMESS-US hamiltonian module is

```
MODULE GAMESS_HAMILTONIAN

CONTAINS
   SUBROUTINE GAMESS_SETUP ( METHOD, BASIS, DFTFUNCTIONAL,   &
                             CHARGE, MULTIPLICITY,           &
                             SELECTION, LINK_ATOM_DISTANCES, &
                             PRINT )
      CHARACTER ( LEN = * ), INTENT(IN)           :: METHOD, BASIS
      CHARACTER ( LEN = * ), INTENT(IN), OPTIONAL :: DFTFUNCTIONAL
      INTEGER, INTENT(IN),               OPTIONAL :: CHARGE, MULTIPLICITY
      LOGICAL, DIMENSION(1:NATOMS), INTENT(IN), OPTIONAL :: SELECTION
      REAL ( KIND = DP ), DIMENSION(:), INTENT(IN), &
                                         OPTIONAL :: LINK_ATOM_DISTANCES
      LOGICAL, INTENT(IN), OPTIONAL               :: PRINT
   END SUBROUTINE GAMESS_SETUP

END MODULE GAMESS_HAMILTONIAN
```

This module specifies the wavefunction or density functional method use for the QM part of the system. The module contains the public subroutine `GAMESS_SETUP`. All the arguments, with the exception of `LINK_ATOM_DISTANCES`, are described in the `GAMESS_DATA` module. The optional `LINK_ATOM_DISTANCES` argument specifies the QM link-atom distances. If not given, the default value is 1.0 Å.

8.4.5 The GAMESS-US Energy Module

The definition of the GAMESS-US energy module is

```
MODULE GAMESS_ENERGY

CONTAINS
   SUBROUTINE ENERGY_GAMESS ( EQM, GRADIENT, PRINT )
      REAL ( KIND = DP ), INTENT(OUT)   :: EQM
      REAL ( KIND = DP ), DIMENSION(1:3,1:NATOMS), INTENT(INOUT), &
                                 OPTIONAL :: GRADIENT
      LOGICAL, INTENT(IN), OPTIONAL     :: PRINT
   END SUBROUTINE ENERGY_GAMESS
END MODULE GAMESS_ENERGY
```

This module contains only the subroutine **ENERGY_GAMESS**. This subroutine makes an operating system (OS) call to execute a script, which requires the `gamess.inp` file in order to run the GAMESS-US program. The subroutine also handles the transfer of the calculated quantum energy from GAMESS-US, and, if necessary, the gradients. The GAMESS-US gradients are calculated by calling the subroutine **GRADIENTS_GAMESS** in the **GAMESS_GRADIENTS** module described next.

8.4.6 The GAMESS-US Gradients Module

The definition of the GAMESS-US gradients module is

```
MODULE GAMESS_GRADIENTS

CONTAINS
   SUBROUTINE GRADIENTS_GAMESS (  GRADIENT )
      REAL ( KIND = DP ), DIMENSION(:,:), INTENT(INOUT) :: GRADIENT
   END SUBROUTINE GRADIENTS_GAMESS

END MODULE GAMESS_GRADIENTS
```

This module returns the calculated GAMESS-US gradients.

8.5 Modifications to DYNAMO Modules

Most of the original DYNAMO modules were left unmodified. Only few modules from the original DYNAMO library distribution have been modified. They are shown in Table 8.2.

Table 8.2: The modified modules in the original DYNAMO distribution.

Module Name	Modification
`POTENTIAL_ENERGY`	subroutine `ENERGY_QUANTUM` is replaced by subroutine `ENERGY_GAMESS`
`ENERGY_NON_BONDING`	function `MOPAC_DATA_ATOM_POSITION` is replaced by function `GAMESS_DATA_ATOM_POSITION`
`DYNAMICS_UTILITIES`	added to subroutine `ENERGY_AND_ACCELERATIONS` calls to subroutines `GAMESS_CONTROLS_WRITE` and `GAMESS_COORDINATES_WRITE`

8.6 An Example of Using DYNAMO/GAMESS-US Library

To reiterate, the creation of the FORTRAN 90 main program to use the DYNAMO library is up to the user. The main program should outline the sequence of steps that should be done in order to achieve the desired simulation outcome. The following `PROGRAM DYNAMOGAMESS` shows an example of how to perform a QM/MM molecular dynamic simulation using the GAMESS-US QM potentials and DYNAMO OPLS force-field.

```
PROGRAM DYNAMOGAMESS

USE DYNAMO       ! . Use the DYNAMO library
IMPLICIT NONE

HAMILTONIAN = "RHF"
BASIS       = "N31 NGAUSS=6"

CALL DYNAMO_HEADER         ! . Print out the header.
CALL TIME_PRINT            ! . Initialization.

! . Process the MM file for bALA.
```

```
CALL MM_FILE_PROCESS ( "bALA.opls_bin", "water.opls" )
! . Construct the system file for bALA.
CALL MM_SYSTEM_CONSTRUCT ( "water.opls_bin", "water.seq" )
! . Save the system file for later use.
CALL MM_SYSTEM_WRITE ( "water.sys_bin" )

CALL COORDINATES_READ ( "water_dimer.crd" )     ! . Read in the coordinates.

! . Define third residue to be QM.
CALL GAMESS_SETUP ( METHOD = HAMILTONIAN, BASIS = BASIS, &
                SELECTION = ATOM_SELECTION ( RESIDUE_NUMBER = (/ 2 /) ), &
                    PRINT = .TRUE.  )

CALL COORDINATES_WRITE        ! . Print the coordinates.
CALL ENERGY                   ! . Calculate initial energy.

! . Initialize the random number seed.
CALL RANDOM_INITIALIZE ( 314171, 314172 )
! . Assign velocities to the atoms at a temperature of 10K.
CALL VELOCITY_ASSIGN ( 10.0_DP, REMOVE_TRANSLATION = .TRUE. )

! . Equilibrate the system.
CALL DYNAMICS_OPTIONS ( TIME_STEP = 0.001_DP, &
                            STEPS = 10000,    &
                         PRINT_FREQUENCY = 100 )
CALL LANGEVIN_VERLET_DYNAMICS ( TBATH = 300.0_DP, GAMMA = 100.0_DP )

! . Run the dynamics.
CALL DYNAMICS_OPTIONS ( TIME_STEP = 0.001_DP, &
                             STEPS = 50000,      &
                        PRINT_FREQUENCY = 100,   &
                         SAVE_FREQUENCY = 100,   &
                        COORDINATE_FILE = "bALA_hpmd1.dcd" )
CALL LANGEVIN_VERLET_DYNAMICS ( TBATH = 300.0_DP, GAMMA = 100.0_DP )

CALL ENERGY                ! . Calculate the final energy.
CALL COORDINATES_WRITE     ! . Print out the final coordinates.
CALL TIME_PRINT            ! . Finish up.

END PROGRAM DYNAMOGAMESS
```

The program begins by invoking **USE DYNAMO** to use the DYNAMO library and writing **IMPLICIT NONE** to force all the varibles to be explicitly typed. Next GAMESS-US method and basis set are defined. This is accomplished by setting the values of the **HAMILTONIAN** and **BASIS**. In this case, the **HAMILTONIAN** is assigned a string value of **RHF**. The **BASIS** is assigned a string value of **N31 NGAUSS=6**, which adheres to how the **$BASIS** group is specified in the GAMESS-US input file (see Figure 8.7). The DYNAMO subroutines can be called appropriately, such as the processing of MM file and reading of coordinates. In order to define the QM part of the system, a call to **GAMESS_SETUP** must be made, passing the values of the **HAMILTONIAN** and **BASIS** to the appropriate arguments **METHOD** and **BASIS**, respectively. Selecting which atoms of the system are assigned QM can be accomplished by assigning values to the **SELECTION** array. In this case, the second residue defined in the `water_dimer.crd` is assigned the QM part. Once the QM part is defined, the rest of the program just calls standard DYNAMO subroutines.

8.7 Conclusion

Knowledge of programming, in particular FORTRAN 90 programming, is necessary in order to utilize the standard DYNAMO library for performing molecular simulations. However, if higher levels of quantum mechanical methods are required for a particular problem, then the DYNAMO library interfaced with the GAMESS-US program as described in this chapter can be utilized.

As in the example given in Section 8.6, the use of the DYNAMO–GAMESS-US program would require assigning values to variables **HAMILTONIAN** and **BASIS**. This is then followed by a call to the **GAMESS_SETUP** subroutine. The rest are just calls to standard DYNAMO subroutines.

Examples of FORTRAN 90 programs invoking the DYNAMO library can be found in the Appendix. These examples are provided so that they can serve as templates which can be easily modified for simulation problems of similar types.

Chapter 9

Summary and Conclusions

Computational chemistry uses the principles of quantum mechanics for theoretical modeling. It is normally associated with molecular modeling, such as molecular mechanics and molecular dynamics simulations. The applications of computational chemistry varies very broadly ranging from simulations of small to large molecules, such as proteins and enzymes.

Large molecular systems require an accurate description of a range of intricate and highly specific inter- and intramolecular interactions. This accurate representation of the interactions can be provided by quantum mechanical (QM) or density functional (DFT) calculations. Although computational power has grown exponentially in the past few years, QM and DFT techniques are still limited to treating molecules of less than 50 atoms. It is true that force-fields for molecular mechanics calculations have been developed and are widely used in simulations of large complex systems, however, they still remain as fixed-charge models. A balance of both the relative speed, comparable to those provided by molecular mechanical calculations, and the incorporation of a dynamical model for the charges, is the driving force for the use of semiempirical methods, especially in a hybrid QM/MM methodology.

In this work, computational tools have been developed for modeling large molecular systems that allow the balance between accuracy of results and speed of computation. The genetic algorithm discussed in Chapter 2 outlined how the GA was implemented and how it can be used for fitting parameters against a set of high quality reference values.

Polysaccharides, which are made up of monosaccharide units, are large molecules and are complex in nature. In order to model polysaccharides by semi-empirical methods, an accurate set of semi-empirical parameters must be used. The genetic algorithm program developed in Chapter 2 was utilized in Chapter 3 in the optimization of semi-empirical PM3 parameters for monosaccharides. The monosaccharides included in the training set are select conformers of glucose, galactose, and mannose. Using the MP2 method, the geometries of these monosaccharides were optimized and relative energies were ranked to serve as a reference for the GA. Next, GA optimizations were performed and a set of semi-empirical parameters which accurately reproduced the reference properties were kept. These parameters can be used in molecular simulations involving monosaccharide and/or polysaccharides.

Chapter 4 discussed the methods that are commonly used to calculate free energies, namely, the method of free energy perturbation and the umbrella sampling. In Chapter 5, the relation between microtubules and the cell and those classes of compounds that affect the microtubules were introduced. Both chapters are necessary in understanding the calculations that were performed in Chapter 6.

The DYNAMO computer program has been extensively used in this thesis. It is a library of FORTRAN 90 modules developed for the simulation of molecular systems using hybrid QM/MM potentials. In Chapter 6, the use of the DYNAMO library has been illustrated in the calculation of free energies of binding of colchicine and its derivatives with different tubulin isoforms, an important component of microtubules. It should be noted, however, that DYNAMO is a library of subroutines designed for molecular simulation and not a standalone program. In order to use it, the user would have to design a series of calls to the modules corresponding to how the user would try to simulate or solve a particular problem. In Chapter 7 and Appendix A, an effort has been made to provide many examples of how to use the library. These examples include the calculation of free energy using umbrella sampling (as in the case of diaminopimelate), commonly used methods such as the addition of hydrogen coordinates and solvation of solutes and protein molecules, and a protocol for simulated annealing.

DYNAMO has many capabilities for molecular simulations. However, its QM potentials are limited to semi-empirical MNDO, AM1, and PM3 potentials. Chapter 8 discussed how the capabilities of the DYNAMO library were extended. FORTRAN 90 modules were written and added to the library to utilize more advanced QM potentials that could be generated by external programs. GAMESS-US is one of the many advanced electronic structure program capable of supplying these potentials and it was used in this work. The modules that were added allowed for accessing the GAMESS-US QM potentials from within DYNAMO simply by calling the appropriate subroutines.

At present, the computational tools developed here would require further validation. For example, the newly obtained set of semi-empirical PM3 parameters for monosaccharides have not yet been tested in molecular dynamics simulation. It will be interesting to use these parameters not only for monosaccharide simulations but also for oligosaccharide simulations as an alternative to MM force fields since semi-empirical methods represent torsional angles much better than MM methods.

Bibliography

[1] Gigant, B.; Curmi, P.; Martin-Barbey, C.; Charbaut, E.; Lachkar, S.; Lebeau, L.; Siavoshian, S.; Sobel, A.; Knossow, M. The 4 Å X-ray structure of a tubulin:stathmin-like domain complex. *Cell* **2000,** *102,* 809-816.

[2] Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry;* Dover Publications, Inc.: Mineola, New York, 1996.

[3] Levine, I. N. *Quantum Chemistry;* Prentice Hall: Upper Saddle River, New Jersey, 5th ed.; 2000.

[4] Jensen, F. *Introduction to Computational Chemistry;* John Wiley & Sons: Chichester, West Sussex, England, 1999.

[5] McQuarrie, D. A. *Quantum Chemistry;* University Science Books: Sausalito, California, 1983.

[6] Roothaan, C. C. J. New developments in molecular orbital theory. *Rev. Mod. Phys.* **1951,** *23,* 69–89.

[7] Hall, G. G. The molecular orbital theory of chemical valency. VIII. A method of calculating ionization potentials. *Proc. R. Soc. A* **1951,** *205,* 541–552.

[8] Raghavachari, K. Electron correlation techniques in quantum chemistry. *Annu. Rev. Phys. Chem.* **1991,** *42,* 615–642.

[9] Saebo, S.; Pulay, P. Local treatment of electron correlation. *Annu. Rev. Phys. Chem.* **1993,** *44,* 213–236.

[10] Raghavachari, K. Electron correlation effects in molecules. *J. Phys. Chem.* **1996,** *100,* 12960–12973.

[11] Bartlett, R. J. Coupled-cluster approach to molecular structure and spectra: A step toward predictive quantum chemistry. *J. Phys. Chem.* **1989,** *93,* 1697-1708.

[12] Møller, C.; Plesset, M. S. Note on an approximation treatment for many-electron systems. *Phys. Rev.* **1934,** *46,* 618–622.

[13] Krishnan, R.; Pople, J. A. Approximate fourth-order perturbation theory of the electron correlation energy. *Int. J. Quantum Chem.* **1978,** *14,* 91–100.

[14] Hohenberg, P.; Kohn, W. Inhomogeneous electron gas. *Phys. Rev.* **1964,** *136,* B864–B871.

[15] Parr, R. G.; Yang, W. *Density-Functional Theory of Atoms and Molecules;* Oxford University Press, Inc.: New York, 1989.

[16] Kohn, W.; Sham, L. J. Self-consistent equations including exchange and correlation effects. *Phys. Rev.* **1965,** *140,* A1133–A1138.

[17] Stowasser, R.; Hoffmann, R. What do the Kohn-Sham orbitals and eigenvalues mean? *J. Am. Chem. Soc.* **1999,** *121,* 3414–3420.

[18] Bouř, P. Comparison of Hartree-Fock and Kohn-Sham determinants as wavefunction. *J. Comput. Chem.* **2000,** *21,* 8–16.

[19] Ceperley, D. M.; Alder, B. J. Ground-state of the electron-gas by a stochastic method. *Phys. Rev. Lett.* **1980,** *45,* 566–569.

[20] Vosko, S. H.; Wilk, L.; Nusair, M. Accurate spin-dependent electron liquid correlation energies for local spin-density calculations – a critical analysis. *Can. J. Phys.* **1980,** *58,* 1200–1211.

[21] Becke, A. D. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A* **1988,** *38,* 3098–3100.

[22] Becke, A. D. Density-functional thermochemistry. IV. A new dynamical correlation functional and implications for exact-exchange mixing. *J. Chem. Phys.* **1996,** *104,* 1040–1046.

[23] Becke, A. D. Density-functional thermochemistry. V. Systematic optimization of exchange-correlation functionals. *J. Chem. Phys.* **1997,** *107,* 8554–8560.

[24] Perdew, J. P. Density-functional approximation for the correlation energy of the inhomogeneous electron gas. *Phys. Rev. B* **1986,** *33,* 8822–8824.

[25] Perdew, J. P.; Wang, Y. Accurate and simple density functional for the electronic exchange energy: Generalized gradient approximation. *Phys. Rev. B* **1986,** *33,* 8800–8802.

[26] Perdew, J. P.; Wang, Y. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B* **1992,** *45*, 13244–13249.

[27] Lee, C.; Yang, W.; Parr, R. G. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B* **1988,** *37,* 785–789.

[28] Becke, A. D. Density-functional thermochemistry: III. The role of exact exchange. *J. Chem. Phys.* **1993,** *98,* 5648-5652.

[29] Huzinaga, S.; Andzelm, J.; Klobukowski, M.; Radzio-Andzelm, E.; Saski, Y.; Tatewaki, H. *Gaussian Basis Sets for Molecular Calculation;* Elsevier: Amsterdam, 1984.

[30] Feller, D.; Davidson, E. R. Basis set selection for molecular calculations. *Chem. Rev.* **1986,** *86*, 681–696.

[31] Feller, D.; Davidson, E. R. Basis Sets for Ab initio Molecular Orbital Calculations and Intermolecular Interactions. In *Reviews in Computational Chemistry*, Vol. 1; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH Publishers, Inc.: New York, 1990.

[32] Pople, J. A.; Santry, D. P.; Segal, G. A. Approximate self-consistent molecular orbital theory. I. Invariant procedures. *J. Chem. Phys.* **1965,** *43,* S129–S135.

[33] Pople, J. A.; Segal, G. A. Approximate self-consistent molecular orbital theory. II. Calculations with complete neglect of differential overlap. *J. Chem. Phys.* **1965,** *43*, S136–S151.

[34] Pople, J. A.; Segal, G. A. Approximate self-consistent molecular orbital theory. III. CNDO results for AB_2 and AB_3 systems. *J. Chem. Phys.* **1966,** *44*, 3289–3296.

[35] Pople, J. A.; Beveridge, D. L.; Dobosh, P. A. Approximate self-consistent molecular orbital theory. V. Intermediate neglect of differential overlap. *J. Chem. Phys.* **1967,** *47*, 2026–2033.

[36] Pople, J. A.; Beveridge, D. L. *Approximate molecular orbital theory;* McGraw-Hill: New York, 1970.

[37] Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. General atomic and molecular electronic structure system. *J. Comput. Chem.* **1993,** *14,* 1347–1363.

[38] Guest, M. F.; Bush, I. J.; van Dam, H. J. J.; Sherwood, P.; Thomas, J. M. H.; van Lenthe, J. H.; Havenith, R. W. A.; Kendrick, J. The GAMESS-UK electronic structure package: Algorithms, developments and applications. *Mol. Phys.* **2005,** *103*, 719–747.

[39] Frisch, M. J. *et al.* "Gaussian 03", Gaussian, Inc., Wallingford, CT, 2004.

[40] Field, M. J.; Albe, M.; Bret, C.; Martin, F.; Thomas, A. The DYNAMO library for molecular simulations using hybrid quantum mechanical and molecular mechanical potentials. *J. Comput. Chem.* **2000,** *21,* 1088–1100.

[41] Stewart, J. J. P. "MOPAC 2007", Stewart Computational Chemistry, Colorado Springs, CO, USA, 2007.

[42] Rocha, G. B.; Freire, R. O.; Simas, A. M.; Stewart, J. P. RM1: A reparameterization of AM1 for H, C, N, O, P, S, F, Cl, Br, and I. *J. Comput. Chem.* **2006,** *27,* 1101–1111.

[43] Stewart, J. P. Optimization of parameters for semiempirical methods V: Modification of nddo approximations and application to 70 elements. *J. Mol. Model.* **2007,** *13,* 1173–1213.

[44] Dewar, M. J. S.; Thiel, W. Ground states of molecules. 38. The MNDO method. Approximations and parameters. *J. Am. Chem. Soc.* **1977,** *99,* 4899–4906.

[45] Dewar, M. J. S.; Thiel, W. Ground states of molecules. 39. MNDO results for molecules containing hydrogen, carbon, nitorgen, and oxygen. *J. Am. Chem. Soc.* **1977,** *99,* 4907–4917.

[46] Thiel, W.; Voityuk, A. A. Extension of the MNDO formalism to *d* orbitals: Integral approximations and preliminary numerical results. *Theor. Chim. Acta.* **1992,** *81,* 391–404.

[47] Thiel, W.; Voityuk, A. A. Extension of MNDO to *d* orbitals: Parameters and results for the halogens. *Int. J. Quantum Chem.* **1992,** *44,* 807–829.

[48] Thiel, W.; Voityuk, A. A. Extension of the MNDO formalism to *d* orbitals: Integral approximations and preliminary numerical results. *Theor. Chim. Acta.* **1996,** *93,* 315.

[49] Thiel, W.; Voityuk, A. A. Extension of MNDO to *d* orbitals: Parameters and results for the second-row elements and for the zinc group. *J. Phys. Chem.* **1996,** *100,* 616–626.

[50] Dewar, M. J. S.; Zoebisch, E. G.; Healy, E. F.; Stewart, J. J. P. AM1: A new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.* **1985,** *107,* 3902–3909.

[51] Dewar, M. J. S.; Dieter, K. M. Evaluation of AM1 calculated proton affinities and deprotonation enthalpies. *J. Am. Chem. Soc.* **1986,** *108,* 8075–8086.

[52] Stewart, J. J. P. Optimization of parameters for semiempirical methods.I. Method. *J. Comput. Chem.* **1989,** *10,* 209–220.

[53] Stewart, J. J. P. Optimization of parameters for semiempirical methods.II. Applications. *J. Comput. Chem.* **1989,** *10,* 221–264.

[54] Allinger, N. L. Conformational analysis. 130. MM2. A hydrocarbon force field utilizing V_1 and V_2 torsional terms. *J. Am. Chem. Soc.* **1977,** *99,* 8127–8134.

[55] Allinger, N. L.; Yuh, Y. H.; Lii, J.-H. Molecular mechanics. The MM3 force field for hydrocarbons. 1. *J. Am. Chem. Soc.* **1989,** *111,* 8551–8566.

[56] Lii, J.-H.; Allinger, N. L. Molecular mechanics. The MM3 force field for hydrocarbons. 2. Vibrational frequencies and thermodynamics. *J. Am. Chem. Soc.* **1989,** *111,* 8566–8575.

[57] Lii, J.-H.; Allinger, N. L. Molecular mechanics. The MM3 force field for hydrocarbons. 3. The van der Waals' potentials and crystal data for aliphatic and aromatic hydrocarbons. *J. Am. Chem. Soc.* **1989,** *111,* 8576–8582.

[58] Allinger, N. L.; Li, F.; Yan, L. Molecular mechanics. The MM3 force field for alkenes. *J. Comput. Chem.* **1990,** *11,* 848–867.

[59] Allinger, N. L.; Li, F.; Yan, L.; Tai, J. C. Molecular mechanics (MM3) calculations on conjugated hydrocarbons. *J. Comput. Chem.* **1990,** *11,* 868–895.

[60] Allinger, N. L.; Chen, K.; Lii, J.-H. An improved force field (MM4) for saturated hydrocarbons. *J. Comput. Chem.* **1996,** *17,* 642–668.

[61] Nevins, N.; Chen, K.; Allinger, N. L. Molecular mechanics (MM4) calculations on alkenes. *J. Comput. Chem.* **1996,** *17,* 669–694.

[62] Nevins, N.; Lii, J.-H.; Allinger, N. L. Molecular mechanics (MM4) calculations on conjugated hydrocarbons. *J. Comput. Chem.* **1996,** *17,* 695–729.

[63] Nevins, N.; Allinger, N. L. Molecular mechanics (MM4) vibrational frequency calculations for alkenes and conjugated hydrocarbons. *J. Comput. Chem.* **1996,** *17*, 730–746.

[64] Allinger, N. L.; Chen, K.; Katzenellenbogen, J. A.; Wilson, S. R.; Anstead, G. M. Hyperconjugative effects on carbon–carbon bond lengths in molecular mechanics (MM4). *J. Comput. Chem.* **1996,** *17*, 747–755.

[65] Weiner, S. J.; Kollman, P. A.; Case, D. A.; Singh, U. C.; Ghio, C.; Alagona, G.; Profeta, S.; Weiner, P. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc.* **1984,** *106*, 765–784.

[66] Cornell, W. D.; Cieplak, P.; Bayly, C. I.; Gould, I. R.; Merz, K. M.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollman, P. A. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.* **1995,** *117*, 5179–5197.

[67] Momany, F. A.; Rone, R. Validation of the general purpose QUANTA® 3.2/CHARMm® force field. *J. Comput. Chem.* **1992,** *13*, 888–900.

[68] Mayo, S. L.; Olafson, B. D.; Goddard III, W. A. DREIDING: A generic force field for molecular simulations. *J. Phys. Chem.* **1990,** *94*, 8897–8909.

[69] Rappé, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard III, W. A.; Skiff, W. M. UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations. *J. Am. Chem. Soc.* **1992,** *114*, 10024–10035.

[70] Casewit, C. J.; Colwell, K. S.; Rappé, A. K. Application of a universal force field to organic molecules. *J. Am. Chem. Soc.* **1992,** *114*, 10035–10046.

[71] Casewit, C. J.; Colwell, K. S.; Rappé, A. K. Application of a universal force field to main group compounds. *J. Am. Chem. Soc.* **1992,** *114*, 10046–10053.

[72] Lifson, S.; Warshel, A. Consistent force field for calculations of conformations, vibrational spectra, and enthalpies of cycloalkane and *n*-alkane molecules. *J. Chem. Phys.* **1968,** *49*, 5116–5129.

[73] Hagler, A. T.; Huler, E.; Lifson, S. Energy functions for peptides and proteins. I. Derivation of a consistent force field including the hydrogen bond from amide crystals. *J. Am. Chem. Soc.* **1974,** *96*, 5319–5327.

[74] Warshel, A.; Lifson, S. Consistent force field calculations. II. Crystal structures, sublimation energies, molecular and lattice vibrations, molecular conformations, and enthalpies of alkanes. *J. Chem. Phys.* **1970,** *53,* 582–594.

[75] Maple, J. R.; Hwang, M. J.; Stockfisch, T. P.; Dinur, U.; Waldman, M.; Ewig, C. S.; Hagler, A. T. Derivation of class II force fields. I. Methodology and quantum force field for the alkyl functional group and alkane molecules. *J. Comput. Chem.* **1994,** *15,* 162–182.

[76] Hwang, M. J.; Stockfisch, T. P.; Hagler, A. T. Derivation of class II force fields. II. Derivation and characterization of a class II force field, CFF93, for the alkyl functional group and alkane molecules. *J. Am. Chem. Soc.* **1994,** *116,* 2515–2525.

[77] Halgren, T. A. Merck molecular force field. I. Basis, form scope, parameterization, and performance of MMFF94. *J. Comput. Chem.* **1996,** *17,* 490–519.

[78] Halgren, T. A. Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions. *J. Comput. Chem.* **1996,** *17,* 520–552.

[79] Halgren, T. A. Merck molecular force field. III. Molecular geometries and vibrational frequencies for MMFF94. *J. Comput. Chem.* **1996,** *17,* 553–586.

[80] Halgren, T. A.; Nachbar, R. B. Merck molecular force field. IV. Conformational energies and geometries for MMFF94. *J. Comput. Chem.* **1996,** *17,* 587–615.

[81] Halgren, T. A. Merck molecular force field. V. Extension of MMFF94 using experimental data, additional computational data, and empirical rules. *J. Comput. Chem.* **1996,** *17,* 616–641.

[82] Halgren, T. A. MMFF VI. MMFF94s option for energy minimization studies. *J. Comput. Chem.* **1999,** *20,* 720–729.

[83] Halgren, T. A. MMFF VII. Characterization of MMFF94, MMFF94s, and other widely available force fields for conformational energies and for intermolecular-interaction energies and geometries. *J. Comput. Chem.* **1999,** *20,* 730–748.

[84] Jorgensen, W. L.; Tirado-Rives, J. The OPLS potential functions for proteins. Energy minimization for crystals of cyclic peptides and crambin. *J. Am. Chem. Soc.* **1988,** *110,* 1657–1666.

[85] Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* **1996,** *118,* 11225–11236.

[86] Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Imprey, R. W.; Klein, M. L. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* **1983,** *79,* 926–935.

[87] Berendsen, H. J. C.; Postma, J. P. M.; von Gunsteren, W. F.; Hermans, J. Interaction models for water in relation to protein hydration. In *Intermolecular forces*; Pullman, B., Ed.; Reidel: Dordrecht, Holland, 1981.

[88] Giao, J. Methods and applications of combined quantum mechanical and molecular mechanical potentials. *Rev. Comput. Chem.* **1996,** *7,* 119-185.

[89] Zhang, Y.; Lee, T.-S.; Yang, W. A pseudobond approach to combining quantum mechanical and molecular mechanical methods. *J. Chem. Phys.* **1999,** *110,* 46–54.

[90] Monard, G.; Loos, M.; Thery, V.; Baka, K.; Rivail, J. Hybrid classical quantum force field for modeling very large molecules. *Int. J. Quantum Chem.* **1996,** *58,* 153–159.

[91] Gao, J.; Amara, P.; Alhambra, C.; Field, M. J. A generalized hybrid orbital (GHO) method for the treatment of boundary atoms in combined QM/MM calculations. *J. Phys. Chem. A* **1998,** *102,* 4714–4721.

[92] Philipp, D. M.; Friesner, R. A. Mixed *ab initio* QM/MM modeling using frozen orbitals and tests with alanine dipeptide and tetrapeptide. *J. Comput. Chem.* **1999,** *20,* 1468–1494.

[93] Singh, U. C.; Kollman, P. A. A combined *ab initio* quantum mechanical and molecular mechanical method for carrying out simulations on complex molecular systems: Applications to the $CH_3Cl + Cl^-$ exchange reaction and gas phase protonation of polyethers. *J. Comput. Chem.* **1986,** *7,* 718–730.

[94] Field, M. J.; Bash, P. A.; Karplus, M. A combined quantum mechanical and molecular mechanical potential for molecular dynamics simulations. *J. Comput. Chem.* **1990,** *11,* 700–733.

[95] Verlet, L. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.* **1967,** *159,* 98–103.

[96] Swope, W. C.; Andersen, H. C.; Berens, P. H.; Wilson, K. R. Computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.* **1982,** *76,* 637–649.

[97] Hockney, R. W. The potential calculation and some applications. In *Methods of Computational Physics*, Vol. 9; Adler, B.; Fernback, S.; Rotenberg, M., Eds.; Academic Press: New York, 1970.

[98] Beeman, D. Some multistep methods for use in molecular dynamics calculations. *J. Comput. Phys.* **1976,** *20,* 130–139.

[99] Berendsen, H. J. C.; Postma, J. P. M.; van Gunsteren, W. F.; DiNola, A.; Haak, J. R. Molecular dynamics with coupling to an external bath. *J. Chem. Phys.* **1984,** *81,* 3684–3690.

[100] Kirkwood, J. G. Theory of solution of molecules containing widely separated charges with special application to zwitterions. *J. Chem. Phys.* **1934,** *2,* 351–361.

[101] Onsager, L. Electric moments of molecules in liquids. *J. Am. Chem. Soc.* **1936,** *58,* 1486–1493.

[102] Miertus, S.; Scrocco, E.; Tomasi, J. Electrostatic interaction of a solute with continuum – A direct utilization of *ab initio* molecular potentials for the provision of solvent effects. *Chem. Phys.* **1981,** *55,* 117–129.

[103] Klamt, A.; Schuurmann, G. COSMO: A new approach to dielectric screening in solvents with explicit expressions for the screening energy and its gradients. *J. Chem. Soc, Perkin Trans.* **1993,** *2,* 799–805.

[104] Klamt, A. Conductor-like screening model for real solvent: A new approach to the quantitative calculation of solvation phenomena. *J. Phys. Chem.* **1995,** *99,* 2224–2235.

[105] Klamt, A.; Jonas, V.; Burger, T.; Lohrenz, J. C. W. Refinements and parameterization of COSMO-RS. *J. Phys. Chem.* **1998,** *102,* 5074–5085.

[106] McGarrah, D. B.; Judson, R. S. Analysis of the genetic algorithm method of molecular conformation determination. *J. Comput. Chem.* **1993,** *14,* 1385–1395.

[107] Judson, R. S.; Jaeger, E. P.; Treasurywala, A. M.; Peterson, M. L. Conformational searching methods for small molecules. II. Genetic algorithm approach. *J. Comput. Chem.* **1993,** *14,* 1407–1414.

[108] Bungay, S. D.; Poirier, R. A.; Charron, R. J. Optimization of transition state structures using genetic algorithms. *J. Math. Chem.* **2000,** *28,* 389–401.

[109] Blommers, M. J. J.; Lucasius, C. B.; Kateman, G.; Kaptein, R. Conformational analysis of a dinucleotide photodimer with the aid of the genetic algorithm. *Biopolymers* **1992,** *32,* 45–52.

[110] Ogata, H.; Akiyama, Y.; Kanehisa, M. A genetic algorithm based molecular modeling technique for RNA stem-loop structure. *Nucleic Acids Res.* **1995,** *23,* 419–426.

[111] Gultyaev, A. P.; van Batenburg, F. H. D.; Pleij, C. W. A. The computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol.* **1995,** *250,* 37–51.

[112] Dandekar, T.; Argos, P. Potential of genetic algorithms in protein folding and protein engineering simulations. *Protein Eng.* **1992,** *5,* 637–645.

[113] Unger, R.; Moult, J. Genetic algorithms for protein folding simulations. *J. Mol. Biol.* **1993,** *231,* 75–81.

[114] Sun, S. Reduced representation model of protein structure prediction: Statistical potential and genetic algorithms. *Protein Sci.* **1993,** *2,* 762–785.

[115] Dandekar, T.; Argos, P. Folding the main chain of small proteins with the genetic algorithm. *J. Mol. Biol.* **1994,** *236,* 844–861.

[116] Sun, S. A genetic algorithm that seeks native states of peptides and proteins. *Biophys. J.* **1995,** *69,* 340–355.

[117] Dandekar, T.; Argos, P. Identifying the tertiary fold of small proteins with different topologies from sequence and secondary structure using the genetic algorithm and extended criteria specific for strand regions. *J. Mol. Biol.* **1996,** *256,* 645–660.

[118] Dandekar, T.; Argos, P. Ab initio tertiary-fold prediction of helical and non-helical protein chains using a genetic algorithm. *Int. J. Biol. Macromol.* **1996,** *18,* 1–4.

[119] Yang, J.-M.; Tsai, C.-H.; Hwang, M.-J.; Tsai, H.-K.; Hwang, J.-K.; Kao, C.-Y. GEM: A Gaussian evolutionary method for predicting protein side-chain conformations. *Protein Sci.* **2002,** *11,* 1897–1907.

[120] Xiao, Y. L.; Williams, D. E. Genetic algorithms for docking of actinomycin D and deoxyguanosine molecules with comparison to the crystal structure of actinomycin D-deoxyguanosine complex. *J. Phys. Chem.* **1994,** *98*,.

[121] Oshiro, C. M.; Kuntz, I. D.; Dixon, J. S. Flexible ligand docking using a genetic algorithm. *J. Comput.-Aided Mol. Design* **1995,** *9*, 113–130.

[122] Goodsell, D. S.; Morris, G. M.; Olson, A. J. Automated docking of flexible ligands: Applications of AutoDock. *J. Mol. Recognition* **1996,** *9*, 1–5.

[123] Holland, J. H. *Adaptation in Natural and Artificial Systems;* University of Michigan Press: Ann Arbor, 1975.

[124] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning;* Addison-Wesley: Reading, MA, 1989.

[125] Davis, L. *Handbook of Genetic Algorithms;* Van Nostrand Reinhold: New York, 1991.

[126] Michalewicz, Z. *Genetic algorithms + Data structures = Evolution Programs;* Springer-Verlag: New York, 1994.

[127] Joines, J.; Houck, C. On the use of non-stationary penalty functions to solve constrained optimization problems with genetic algorithms. In *IEEE World Congress on Evolutionary Computation*; IEEE: Orlando, FL., 1994.

[128] Gosling, J.; Joy, B.; Steele, G.; Bracha, G. *The Java language specification;* Addison-Wesley: Boston, 3rd ed.; 2005.

[129] Glennon, T. M.; Zheng, Y.-J.; Le Grand, S. M.; Shutzberg, B. A.; Merz, K. M. A force field for monosaccharides and (1→4) linked polysaccharides. *J. Comput. Chem.* **1994,** *15*, 1019–1040.

[130] Woods, R. J.; Dwek, R. A.; Edge, C. J.; Fraser-Reid, B. Molecular mechanical and molecular dynamical simulations of glycoproteins and oligosaccharides.1. GLYCAM_93 parameter development. *J. Phys. Chem.* **1995,** *99*, 3832–3846.

[131] Senderowitz, H.; Parish, C.; Still, W. C. Carbohydrates: United atom AMBER parameterization of pyranoses and simulations yielding anomeric free energies. *J. Am. Chem. Soc.* **1996,** *118*, 2078–2086.

[132] Momany, F. A.; Willett, J. L. Computational studies on carbohydrates: in vacuo studies using a revised AMBER force field, AMB99C, designed for α-(1$\rightarrow$) linkages. *Carbohydr Res.* **2000,** *326,* 194–209.

[133] Ha, S. N.; Giammona, A.; Field, M.; Brady, J. W. A revised potential-energy surface for molecular mechanics studies of carbohydrates. *Carbohydr Res.* **1988,** *180,* 207–221.

[134] Reiling, S.; Schlenkrich, M.; Brickmann, J. Force field parameters for carbohydrates. *J. Comput. Chem.* **1996,** *17,* 450–468.

[135] Ott, K. H.; Meyer, B. Parameterization of GROMOS force-field for oligosaccharides and assessment of efficientcy of molecular dynamics simulations. *J. Comput. Chem.* **1996,** *17,* 1068–1084.

[136] Stewart, J. J. P. Optimization of parameters for semiempirical methods. III. Extension of PM3 to Be, Mg, Zn, Ga, Ge, As, Se, Cd, In, Sn, Sb, Te, Hg, Tl, Pb, and Bi. *J. Comput. Chem.* **1991,** *12,* 320–341.

[137] Stewart, J. J. P. Optimization of parameters for semiempirical methods. IV. Extension of MNDO, AM1, and PM3 to more main group elements. *J. Mol. Model.* **2004,** *10,* 155–164.

[138] Thiel, W.; Voityuk, A. A. Extension of the MNDO formalism to *d* orbitals: Integral approximations and preliminary numerical results. *Theor. Chim. Acta.* **1996,** *93,* 315.

[139] Thiel, W.; Voityuk, A. A. Extension of the MNDO formalism to *d* orbitals: Integral approximations and preliminary numerical results. *Theor. Chim. Acta.* **1996,** *93,* 315.

[140] Hajduk, P. J.; Horita, D. A.; Lerner, L. E. Picosecond dynamics of simple monosaccharides as probed by NMR and molecular dynamics simulations. *J. Am. Chem. Soc.* **1993,** *115,* 9196–9201.

[141] Rossi, I.; Truhlar, D. G. Parameterization of NDDO wavefunctions using genetic algorithms. An evolutionary approach to parameterizing potential energy surfaces and direct dynamics calculations for organic reactions. *Chem. Phys. Lett.* **1995,** *233,* 231–236.

[142] Kony, D.; Damm, W.; Stoll, S.; Gunsteren, W. F. An improved OPLS-AA force field for carbohydrates. *J. Comput. Chem.* **2002,** *23,* 1416–1429.

[143] Dunning Jr., T. H. Gaussian basis sets for use in correlated molecular calculations. I. the atoms boron through neon and hydrogen. *J. Chem. Phys.* **1989,** *90,* 1007–1023.

[144] Kabsch, W. A solution for the best rotation to relate two sets of vectors. *Acta Cryst.* **1976,** *A32,* 922–923.

[145] Karplus, M.; McCammon, J. A. Dynamics of proteins: Elements and function. *Annu. Rev. Biochem.* **1983,** *53,* 263–300.

[146] Jorgensen, W. L. Theoretical studies of medium effects on conformational equilibria. *J. Phys. Chem.* **1983,** *87,* 5304–5314.

[147] Pearlman, D. A.; Kollman, P. A. The overlooked bond-stretching contribution in free energy perturbation calculations. *J. Chem. Phys.* **1991,** *94,* 4532–4545.

[148] Mitchell, M. J.; McCammon, J. A. Free energy difference calculations by thermodynamic integration: Difficulties in obtaining a precise value. *J. Comput. Chem.* **1991,** *12,* 271–275.

[149] Hodel, A.; Simonson, T.; Fox, R. O.; Brunger, A. T. Conformational substrates and uncertainty in macromolecular free energy calculations. *J. Phys. Chem.* **1993,** *97,* 3409-3417.

[150] Pearlman, D. A. Determining the contributions of constraints in free energy calculations: Development, characterization, and recommendations. *J. Chem. Phys.* **1993,** *98,* 8946–8957.

[151] Pearlman, D. A. Free energy derivatives: A new method for probing the convergence problem in free energy calculations. *J. Comput. Chem.* **1994,** *15,* 105–123.

[152] Helms, V.; Wade, R. C. Free energies of hydration from thermodynamic integration: Comparison of molecular mechanics force fields and evaluation of calculation accuracy. *J. Comput. Chem.* **1997,** *18,* 449–462.

[153] Zwanzig, R. W. High-temperature equation of state by a perturbation method. I. Nonpolar gases. *J. Chem. Phys.* **1954,** *22,* 1420–1426.

[154] Torrie, G. M.; Valleau, J. P. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *J. Comput. Phys.* **1977,** *23,* 187–199.

[155] Kumar, S.; Bouzida, D.; Swendsen, R. H.; Kollman, P. A.; Rosenberg, J. M. The weighted histogram analysis method for free-energy calculations on biomolecules. I. The method. *J. Comput. Chem.* **1992,** *13,* 1011–1021.

[156] Oosawa, F.; Asakura, S. *Thermodynamics of the Polymerization of Protein;* Academic Press: London, 1975.

[157] Wegner, A.; Engel, J. Kinetics of the cooperative association of actin to actin filaments. *Biophys. Chem.* **1975,** *3,* 215–225.

[158] Korn, E. D. Actin polymerization and its regulation by proteins from nonmuscle cells. *Physiol. Rev.* **1982,** *62,* 672–737.

[159] Pollard, T. D.; Craig, S. W. Mechanism of actin polymerization. *Trends Biochem. Sci.* **1982,** *7,* 55–58.

[160] Stryer, L. *Biochemistry, 3rd ed.;* W. H. Freeman: New York, 1988.

[161] Frieden, C.; Goddette, D. W. Polymerization of actin and actin-like systems: Eevaluation of the time course of polymerization in relation to the mechanism. *Biochemistry* **1983,** *22,* 5836–5843.

[162] Hamel, E. *Microtubule Proteins;* CRC Press: Boca Raton, FL, 1990.

[163] Wilson, L.; Jordan, M. *Microtubules;* Wiley-Liss: New York, 1994.

[164] Huzil, J.; Ludueña, R.; Tuszynski, J. Comparative modelling of human β tubulin isotypes and implications for drug binding. *Nanotechnology* **2006,** *17,* S90-S100.

[165] Panda, D.; Miller, H.; Banerjee, A.; Ludueña, R.; Wilson, L. Microtubule dynamics in vitro are regulated by the tubulin isotype composition. *Proc. Natl. Acad. Sci. USA* **1994,** *91,* 11358-11362.

[166] Banerjee, A.; Kasmala, L. Differential assembly kinetics of α-tubulin isoforms in the presence of paclitaxel. *Biochem. Biophys. Res. Commun.* **1998,** *245,* 349-351.

[167] Sept, D.; Baker, N.; McCammon, J. The physical basis of microtubule structure and stability. *Protein Sci.* **2003,** *12,* 2257-2261.

[168] Ranganathan, S.; Dexter, D.; Benetatos, C.; Chapman, A.; Tew, K.; Hudes, G. Increase of βIII- and βIVa-tubulin isotopes in human prostate carcinoma cells as a result of estramustine resistance. *Cancer Res.* **1996,** *56,* 2584-2589.

[169] Liu, B.; Staren, E.; Iwamura, T.; Appert, H.; Howard, J. Mechanisms of taxotere-related drug resistance in pancreatic carcinoma. *J. Surg. Res.* **2001,** *99,* 179-186.

[170] Hari, M.; Yang, H.; Zeng, C.; Canizales, M.; Cabral, F. Expression of class III β-tubulin reduces microtubule assembly and confers resistance to paclitaxel. *Cell Motil. Cytoskeleton* **2003,** *56,* 45-56.

[171] Mozzetti, S.; Ferlini, C.; Concolino, P.; Filippetti, F.; Raspaglio, G.; Prislei, S.; Gallo, D.; Martinelli, E.; Ranelletti, F.; Ferrandina, G.; Scambia, G. Class III β-tubulin overexpression is a prominent mechanism of paclitaxel resistance in ovarian cancer patients. *Clin. Cancer Res.* **2005,** *11,* 298-305.

[172] Khan, I.; Ludueña, R. Different effects of vinblastine on the polymerization of isotypically purified tubulins from bovine brain. *Invest. New Drugs* **2003,** *21,* 3-13.

[173] Banerjee, A.; Ludueña, R. Kinetics of colchicine binding to purified β-tubulin isotypes from bovine brain. *J. Biol. Chem.* **1992,** *267,* 13335-13339.

[174] Hardman, J.; Limbird, L. *Goodman & Gilman's The Pharmaco-logical basis of therapeutics;* McGraw-Hill: New York, 9th ed.; 1996.

[175] Scott, C.; Walker, C.; Neal, D.; Harper, C.; Bloodgood, R.; Somers, K.; Mills, S.; Rebhun, L.; Levine, P. β-tubulin epitope expression in normal and malignant epithelial cells. *Arch. Otolaryngol. Head Neck Surg.* **1990,** *116,* 583-589.

[176] Apweiler, R.; Bairoch, A.; Wu, C.; Barker, W.; Boeckmann, B.; Ferro, S.; Gasteiger, E.; Huang, H.; Lopez, R.; Magrane, M.; Martin, M.; Natale, D.; O'Donovan, C.; Redaschi, N.; Yeh, L. UniProt: The universal protein knowledgebase. *Nucleic Acids Res.* **2004,** *32,* D115–D119.

[177] Thompson, J.; Higgins, D.; Gibson, T. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.* **1994,** *22,* 4673–4680.

[178] Lee, M.; Lewis, S.; Wilde, C.; Cowan, N. Evolutionary history of a multigene family: An expressed human β-tubulin gene and three processed pseudogenes. *Cell* **1983,** *33,* 477-487.

[179] Hall, J.; Dudley, L.; Dobner, P.; Lewis, S.; Cowan, N. Identification of two human β-tubulin isotypes. *Mol. Cell Biol.* **1983,** *3,* 854-862.

[180] Lewis, S.; Gilmartin, M.; Hall, J.; Cowan, N. Three expressed sequences within the human β-tubulin multigene family each define a distinct isotype. *J. Mol. Biol.* **1985,** *182,* 11-20.

[181] Burgoyne, R. D.; Cambray-Deakin, M.; Lewis, S.; Sarkar, S.; Cowan, N. Differential distribution of β-tubulin isotypes in cerebellum. *EMBO J.* **1988,** *7,* 2311–2319.

[182] Strausberg, R.; Team, T. M. G. C. P. Generation and initial analysis of more than 15,000 full-length human and mouse cDNA sequences. *Proc. Natl. Acad. Sci. USA* **2002,** *99,* 16899-16903.

[183] Chothia, C.; Lesk, A. The relation between the divergence of sequence and structure in proteins. *EMBO J.* **1986,** *5,* 823-826.

[184] Luthy, R.; Bowie, J.; Eisenberg, D. Assessment of protein models with three-dimensional profiles. *Nature* **1992,** *356,* 83-85.

[185] Nogales, E.; Wolf, S.; Downing, K. Structure of the α/β tubulin dimer by electron crystallography. *Nature* **1998,** *391,* 199-203.

[186] Lowe, J.; Li, H.; Downing, K.; Nogales, E. Refined structure of α/β at 3.5 Å resolution. *J. Mol. Biol.* **2001,** *313,* 1045-1057.

[187] Gigant, B.; Wang, C.; Ravelli, R.; Roussi, F.; Steinmetz, M.; Curmi, P.; Sobel, A.; Knossow, M. Structural basis for the regulation of tubulin by vinblastine. *Nature* **2005,** *435,* 519-522.

[188] Sali, A.; Blundell, T. Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* **1993,** *234,* 779-815.

[189] Sanchez, R.; Sali, A. Comparative protein structure modeling: Introduction and practical examples with Modeller. *Methods Mol. Biol.* **2000,** *143,* 97-129.

[190] Hooft, R.; Vriend, G.; Sander, C.; Abola, E. Errors in protein structures. *Nature* **1996,** *381,* 272.

[191] Laskowski, R.; MacArthur, M.; Moss, D.; Thornton, J. Procheck: A program to check the stereochemical quality of protein structures. *J. Appl. Cryst.* **1993,** *26,* 283-291.

[192] Mitchison, T.; Kirschner, M. Dynamic instability of microtubule growth. *Nature* **1984,** *312,* 237-242.

[193] Toso, R.; Jordan, M.; Farrell, K.; Matsumoto, B.; Wilson, L. Kinetic stabilization of microtubule dynamic instability in vitro by vinblastine. *Biochemistry* **1993,** *32,* 1285-1293.

[194] Ravelli, R.; Gigant, B.; Curmi, P.; Jourdain, I.; Lachkar, S.; Sobel, A.; Knossow, M. Insight into tubulin regulation from a complex with colchicine and a stathmin-like domain. *Nature* **2004,** *428,* 198-202.

[195] Ludueña, R.; Roach, M. Tubulin sulfhydryl groups as probes and targets for antimitotic and antimicrotubule agents. *Pharmacol. Ther.* **1991,** *49,* 133-152.

[196] Bai, R.; Covell, D.; Pei, X.; Ewell, J.; Nguyen, N.; Brossi, A.; Hamel, E. Mapping the binding site of colchicinoids on β-tubulin: 2-chloroacetyl-2-demethylthiocolchicine covalently reacts predominantly with cysteine 239 and secondarily with cysteine 354. *J. Biol. Chem.* **2000,** *275,* 40443-40452.

[197] Chaudhuri, A.; Seetharamalu, P.; Schwarz, P.; Hausheer, F.; Ludueña, R. The interaction of the B-ring of colchicine with α-tubulin: A novel footprinting approach. *J. Mol. Biol.* **2000,** *303,* 679-692.

[198] Ravelli, R.; Gigant, B.; Curmi, P.; Jourdain, I.; Lachkar, S.; Sobel, A.; Knossow, M. Insight into tubulin regulation from a complex with colchicine and a stathmin-like domain. *Nature* **2004,** *428,*.

[199] Huzil, J.; Chen, K.; Kurgan, L.; Tuszynski, J. The roles of $\beta-$tubulin mutations and isotype expression in acquired drug resistance. *Cancer Informatics* **2007,** *3,* 159-181.

[200] Schaftenaar, G.; Noordik, J. Molden: A pre- and post-processing program for molecular and electronic structures. *J. Comput.-Aided Mol. Design* **2000,** *14,* 123-134.

[201] Dewar, M.; Zoebisch, E.; Healy, E.; Stewart, J. AM1: A new general purpose quantum mechanical molecular model. *J. Am. Chem. Soc.* **1985,** *107,* 3902-3909.

[202] Dewar, M.; Dieter, K. Evaluation of AM1 calculated proton affinities and deprotonation enthalpies. *J. Am. Chem. Soc.* **1986,** *108,* 8075-8086.

[203] Woodcock, H. L.; Hodošček, M.; Brooks, B. R. Exploring scc-dftb paths for mapping qm/mm reaction mechanisms. *J. Phys. Chem. A* **2007,** *111,* 5720-5728.

[204] Jorgensen, W.; Maxwell, D.; Tirado-Rives, J. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* **1996,** *118,* 11225-11236.

[205] Jorgensen, W.; Chandrasekhar, J.; Madura, J.; Impey, R.; Klein, M. Comparison of simple potential functions for simulating liquid water. *J. Chem. Phys.* **1983,** *79,* 926-935.

[206] Tembe, B.; McCammon, J. Ligand-receptor interactions. *Computers & Chemistry* **1984,** *8,* 281-283.

[207] Sigfridsson, E.; Ryde, U. Comparison of methods for deriving atomic charges from the electrostatic potential and moments. *J. Comput. Chem.* **1998,** *19,* 377-395.

[208] Jordan, M.; Wilson, L. Microtubules as a target for anticancer drugs. *Nat. Rev. Cancer* **2004,** *4*, 253-265.

[209] Bugg, T. D. H. Bacterial peptidoglycan biosynthesis and its inhibition. *Compr. Natural Prod. Chem.* **1999,** *3*, 241–294.

[210] Koo, C. W.; Sutherland, A.; Vederas, J. C.; Blanchard, J. S. Identification of active site cysteine residues that function as general bases: diaminopimelate epimerase. *J. Am. Chem. Soc.* **2000,** *122*, 6122–6123.

[211] Pillai, B.; Cherney, M. M.; Diaper, C. M.; Sutherland, A.; Blanchard, J. S.; Vederas, J. C.; James, M. N. Structural insights into stereochemical inversion by diaminopimelate epimerase: An antibacterial drug target. *Proc. Natl. Acad. Sci. USA* **2006,** *103*, 8668–8673.

[212] Field, M. J. The pDynamo program for molecular simulations using hybrid quantum chemical and molecular mechanical potentials. *J. Chem. Theory Comput.* **2008,** *4*, 1151–1161.

[213] Metcalf, M.; Reid, J. FORTRAN 90 *Explained;* Oxford University Press, Inc.: New York, 1990.

[214] Field, M. J. *A Practical Introduction to the Simulation of Molecular Systems;* Cambridge University Press: Cambridge, 1999.

Appendix A

Miscellaneous Topics: Examples for Using Dynamo Library

The Fortran 90 programs that are provided here can serve as templates for performing molecular simulations using the Dynamo library. They can be modified and applied to simulation problems of similar in nature.

A.1 Addition of hydrogen coordinates: An example using α/β-tubulin coordinates

Figure A.1 lists the program used to build the hydrogen coordinates from a coordinate file that only contains heavy atoms. The program require three files: `oplsa`, `ab_tub_I_noh.crd` and `ab_tub_I.seq`. The `oplsa` contains the OPLS-AA force field definition for each of the residues that make up the syatem. The `ab_tub_I_noh.crd` file contains coordinates of heavy atoms of the α/βI-tubulin isoform while the `ab_tub_I.seq` file is necessary in order to construct the system. In Figure A.2, a truncated version of the sequence file is shown. Contained in this file is the declaration of the number of sequences which corresponds to the two subsystems making up the α/βI-tubulin: `SUBSYSTEM AB_TUB_I_SS1`–the α-tubulin consisting of 436 amino acid residues, and `SUBSYSTEM AB_TUB_I_SS2`–the βI-tubulin consisting of 428 amino acid residues. A call to subroutine `BUILD_HYDROGENS` determines which heavy atoms requires the addition of hydrogen and how many hydrogen atoms are required per heavy atom in order to satisfy the shell valency. The coordinates with the hydrogens are then written to the file `ab_tub_I.crd` by making to the subroutine `COORDINATES_WRITE()`.

```
PROGRAM HBUILD

USE DYNAMO
IMPLICIT NONE

CALL DYNAMO_HEADER
CALL TIME_PRINT

CALL MM_FILE_PROCESS     ( "ab_tub_I.opls_bin", "oplsaa" )
CALL MM_SYSTEM_CONSTRUCT ( "ab_tub_I.opls_bin", "ab_tub_I.seq" )

CALL COORDINATES_READ    ( "ab_tub_I_noh.crd" )
CALL BUILD_HYDROGENS

CALL COORDINATES_WRITE ( "ab_tub_I.crd" )

CALL TIME_PRINT
END PROGRAM HBUILD
```

Figure A.1: Program for building the hydrogen coordinates.

A.2 Solvating the system: An example of solvating colchicine-α/β-tubulin system

In working with biomolecular systems, solvation of the protein or enzyme with the solvent, such as water, is important. An example of a program for solvating the colchicine-α/β-tubulin system is shown in Figure A.3. In this example, a 115 Å×75 Å×75 Å rectangular box of water molecules is used and is superimposed on the colchicine-α/β-tubulin system.

The program begins by processing the force field for the solvent (`oplsaa`) and then constructing the system using the `waterbox_1157575.seq` file. Similarly, the same process can be done for the colchicine-α/β-tubulin. In this case, a previously saved file `ovc_d00-ab_tub_I.sys_bin` is used instead. This file is read by making a call to `MM_SYSTEM_READ()`. The information contained

```
SEQUENCE
2
SUBSYSTEM  AB_TUB_I_SS1
436
  ARG ;  GLU ;  CYS ;  ILE ;  SER ;  ILE ;  HIS ;  VAL ;  GLY ;  GLN
  ALA ;  GLY ;  VAL ;  GLN ;  ILE ;  GLY ;  ASN ;  ALA ;  CYS ;  TRP
  .
  .
  .
  ARG ;  GLU ;  ASP ;  MET ;  ALA ;  ALA ;  LEU ;  GLU ;  LYS ;  ASP
  TYR ;  GLU ;  GLU ;  VAL ;  GLY ;  VAL
Variant N_TERMINAL  ARG  1
Variant C_TERMINAL  VAL  436
END

SUBSYSTEM  AB_TUB_I_SS2
428
  MET ;  ARG ;  GLU ;  ILE ;  VAL ;  HIS ;  ILE ;  GLN ;  ALA ;  GLY
  GLN ;  CYS ;  GLY ;  ASN ;  GLN ;  ILE ;  GLY ;  ALA ;  LYS ;  PHE
  .
  .
  .
  ALA ;  GLU ;  SER ;  ASN ;  MET ;  ASN ;  ASP ;  LEU ;  VAL ;  SER
  GLU ;  TYR ;  GLN ;  GLN ;  TYR ;  GLN ;  ASP ;  ALA
Variant N_TERMINAL  MET  1
Variant C_TERMINAL  ALA  428
END

END
```

Figure A.2: A portion of the `ab_tub_I.seq` file.

in this file such as `NATOMS` (the total number of atoms) and `NRESID` (the total number of residues) is saved for later use.

The rectangular box of water is then superimposed on the colchicine-α/β-tubulin system. Water molecules in the box within 2.5 Å `BUFFER`, which are overlapping with the colchicine-α/β-tubulin, were deleted. This is accomplished by assigning correct values for the logical variable `FLAG`. And finally, the water-solvated colchicine-α/β-tubulin system is saved and coordinates are written to a file.

A.3 A molecular dynamics simulation: Langevin dynamics

One way of calculating free energy via molecular dynamics is by using the Langevin algorithm. An example of performing Langevin molecular dynamics simulation is given in Figure A.4. Again, the program requires setting up the molecular mechanics force field of the system via a series of subroutine calls to `MM_FILE_PROCESS()`, `MM_SYSTEM_CONSTRUCT()`, and `MM_SYSTEM_WRITE/READ()`. This is then followed by reading in the coordinates.

In order not to calculate all non-bonding interactions in the system, the non-bonding energy options are setup by giving values to the `LIST_CUTOFF` and the `OUTER_CUTOFF` and `INNER_CUTOFF` in the `ENERGY_NON_BONDING_OPTIONS()` subroutine.

To give motion to the atoms in the system, the velocities of are assigned by passing the required temperature to `VELOCITY_ASSIGN()`. The system is then equilibrated for a specified number of steps before the actual dynamics is performed. It should be noted that the options for performing the molecular dynamics are set up by passing the corresponding arguments to `DYNAMICS_OPTIONS()`. The Langevin molecular dynamics algorithm, `LANGEVIN_VERLET_DYNAMICS()`, is invoked by giving it two arguments, `TBATH` (the temperature of the external bath), and `GAMMA` (the friction coefficient).

```
PROGRAM ADD_SOLVENT
USE DYNAMO
IMPLICIT NONE

! Distance of water OXYGEN from atoms of the solute
REAL ( KIND = DP ), PARAMETER :: BUFFER = 2.5_DP

INTEGER :: I, IRES, NSATOMS, NSRESID, O
LOGICAL, ALLOCATABLE, DIMENSION(:) :: FLAG

CALL DYNAMO_HEADER

! Process the solvent
CALL MM_FILE_PROCESS      ( "solvent.opls_bin", "oplsaa" )
CALL MM_SYSTEM_CONSTRUCT ( "solvent.opls_bin", "waterbox_1157575.seq" )
CALL MM_SYSTEM_WRITE      ( "waterbox_1157575.sys_bin" )

! Read and save the number of solute atoms and residues.
CALL MM_SYSTEM_READ ( "ovc_d00-ab_tub_I.sys_bin" )
NSATOMS = NATOMS;   NSRESID = NRESID

! Append the system file for the box of waters.
CALL MM_SYSTEM_APPEND     ( "waterbox_1157575.sys_bin" )

! Allocate the FLAG array and select the solute atoms.
ALLOCATE ( FLAG(1:NATOMS) ) ; FLAG = .FALSE. ; FLAG(1:NSATOMS) = .TRUE. .

CALL COORDINATES_READ ( "ovc_d00-ab_tub_I.crd", SELECTION = FLAG )
CALL COORDINATES_READ ( "waterbox_1157575.crd", SELECTION = .NOT. FLAG )

FLAG = .FALSE.                  ! Reinitialize the FLAG array.
DO IRES = (NSRESID+1),NRESID    ! Loop over the solvent residues.
   O = RESIND(IRES)+1           ! Find the oxygen in the residue.
   DO I = 1,NSATOMS     ! Find the minimum distance to the solute atoms.
         IF ( SUM ( ( ATMCRD(1:3,I) - ATMCRD(1:3,O) )**2 ) <= BUFFER**2 ) THEN
            FLAG(RESIND(IRES)+1:RESIND(IRES+1)) = .TRUE. ; EXIT
         END IF
   END DO
END DO

! Remove all the water molecules overlapping with the atoms of the
! protein-ions to create the solvated protein-ions in the box.
CALL MM_SYSTEM_DELETE ( FLAG )

CALL MM_SYSTEM_WRITE    ( "ovc_d00-ab_tub_I_water.sys_bin" )
CALL COORDINATES_WRITE ( "ovc_d00-ab_tub_I_water.crd" )

DEALLOCATE ( FLAG )
END PROGRAM ADD_SOLVENT
```

Figure A.3: Program for solvating the colchicine-α/β-tubulin complex in a rectangular box of water molecules.

```
PROGRAM LANGEVIN

USE DYNAMO
IMPLICIT NONE

INTEGER, PARAMETER  :: RND = 681751
INTEGER, PARAMETER  :: PRINT_FREQ = 100, SAVE_FREQ = 100

! Non-bonding cut-offs
REAL ( KIND = DP ), PARAMETER :: LIST_CO  = 15.0_DP
REAL ( KIND = DP ), PARAMETER :: OUTER_CO = 13.0_DP
REAL ( KIND = DP ), PARAMETER :: INNER_CO =  9.0_DP

REAL ( KIND = DP ), PARAMETER :: MD_TIME_STEP = 0.001_DP

CALL DYNAMO_HEADER
CALL TIME_PRINT

CALL MM_FILE_PROCESS     ( "ovc_d00-ab_tub_I-water.opls_bin", "oplsaa" )
CALL MM_SYSTEM_CONSTRUCT ( "ovc_d00-ab_tub_I-water.opls_bin", &
                           "ovc_d00-ab_tub_I-water.seq" )
CALL MM_SYSTEM_WRITE     ( "ovc_d00-ab_tub_I-water.sys_bin" )
CALL MM_SYSTEM_READ      ( "ovc_d00-ab_tub_I-water.sys_bin" )
CALL COORDINATES_READ    ( "ovc_d00-ab_tub_I-water_cg.crd" )
CALL ENERGY_NON_BONDING_OPTIONS ( LIST_CUTOFF  =  LIST_CO,  &
                                  OUTER_CUTOFF = OUTER_CO,  &
                                  INNER_CUTOFF = INNER_CO   )
CALL RANDOM_INITIALIZE ( RND, RND+1 )
CALL VELOCITY_ASSIGN ( TNEEDED = 300.0_DP, REMOVE_TRANSLATION = .FALSE. )

! Equilibrate the system for 10 ps.
CALL DYNAMICS_OPTIONS ( TIME_STEP = MD_TIME_STEP,        &
                            STEPS = 10000,    &
                  PRINT_FREQUENCY = PRINT_FREQ           )
CALL LANGEVIN_VERLET_DYNAMICS ( TBATH = 300.0_DP, GAMMA = 100.0_DP )

! Run the dynamics for 100 ps.
CALL DYNAMICS_OPTIONS ( TIME_STEP = MD_TIME_STEP,        &
                            STEPS = 100000,              &
                  PRINT_FREQUENCY = PRINT_FREQ,          &
                   SAVE_FREQUENCY = SAVE_FREQ,           &
                  COORDINATE_FILE = "ovc_d00-ab_tub_I-water.dcd" )
CALL LANGEVIN_VERLET_DYNAMICS ( TBATH = 300.0_DP, GAMMA = 100.0_DP )

CALL TIME_PRINT
END PROGRAM LANGEVIN
```

Figure A.4: Program for performing Langevin molecular dynamics simulation.

A.4 A protocol for simulated annealing: An application to metal–rare gas clusters

In the method of simulated annealing, the system is heated up to a high temperature then followed by controlled or gradual cooling in the hope of locating a good approximation to the global optimum of a given potential energy landscape. The simulated annealing method is appropriate for those systems that have multiple local minima, such as rare gas clusters and metal-rare gas clusters.

The DYNAMO library can be used to perform simulated annealing. The following program listed in Figures A.5 and A.6 shows an example of performing simulated annealing on metal-rare gas clusters. The cluster consists of one strontium atom and 24 helium atoms. The program listing is similar to that given in the book by Field [214] and uses the same subroutines such as `ADD_CONSTRAINTS`, `STRUCTURE`, and `WRITE_RESULTS`. The subroutine `WRITE_STRUCTURE` writes to a file the coordinates of the cluster (see Figure A.7).

In order to set up the system, however, the OPLS force field and the sequence file must be defined. Figures A.8 and A.9 shows the contents of the `M_He.opls` and `sr_he_24.seq` files, respectively. Since in the `M_He.opls` file the force field for other Group II elements is defined, the same OPLS file can also be used for those elements.

After executing the program, out of the 100 trials or simulated annealing cycles, the lowest annealed energy is -9.353 kJ/mol with a frequency count of 24 out of 100, the rest being higher-energy local minima.

```
PROGRAM SR_HE_24
USE DYNAMO
IMPLICIT NONE

INTEGER, PARAMETER :: NTRIES = 100, SEED0 = 100000
INTEGER            :: ITRY, TMPBND, IEUNIT, MEUNIT, AEUNIT, FEUNIT
REAL ( KIND = DP ) :: TSTART
REAL ( KIND = DP ), DIMENSION(1:NTRIES) :: PE0, PE1, PE2, PE3
REAL ( KIND = DP ), ALLOCATABLE, DIMENSION(:,:) :: TMPCRD

CALL DYNAMO_HEADER
CALL TIME_PRINT
CALL MM_FILE_PROCESS     ( "sr_he_24.opls_bin", "M_He.opls" )
CALL MM_SYSTEM_CONSTRUCT ( "sr_he_24.opls_bin", "sr_he_24.seq" )

CALL ADD_CONSTRAINTS

ALLOCATE ( TMPCRD(1:3,1:NATOMS) )
IEUNIT = NEXT_UNIT ( ) ; MEUNIT = NEXT_UNIT ( )
AEUNIT = NEXT_UNIT ( ) ; FEUNIT = NEXT_UNIT ( )

DO ITRY = 1,NTRIES
   NBONDS = TMPBND

   ! Generate the coordinates for the cluster atoms.
   CALL STRUCTURE

   ! Initialize the coordinates.
   ATMCRD = TMPCRD
   CALL WRITE_STRUCTURE ( "sr_he_24_ie.xyz", IEUNIT, &
                          TEMPERATURE = TSTART )

   ! Do a short dynamics at TSTART to remove constraints in the system.
   CALL VELOCITY_ASSIGN ( TSTART, PRINT = .FALSE. )
   CALL DYNAMICS_OPTIONS ( 0.001_DP, 1000, 0 )
   CALL VELOCITY_VERLET_DYNAMICS ( TARGET_TEMPERATURE = TSTART, &
                                      SCALE_FREQUENCY = 1,      &
                                      SCALE_OPTION = "CONSTANT" )
   ! Calculate the starting energy.
   CALL ENERGY ( PRINT = .FALSE. ) ; PE0(ITRY) = ETOTAL

   ! Save the coordinates.
   TMPCRD = ATMCRD
```

Figure A.5: Program for performing a simulated annealing on a cluster of one strontium and 24 helium atoms.

```
      ! Directly minimize the coordinates (with constraints).
      CALL OPTIMIZE_CONJUGATE_GRADIENT ( GRADIENT_TOLERANCE = 1.0E-4_DP, &
                                            PRINT_FREQUENCY = 0,         &
                                            STEP_NUMBER = 100000 )
      ! Remove the constraints and reminimize.
      NBONDS = 0
      CALL OPTIMIZE_CONJUGATE_GRADIENT ( GRADIENT_TOLERANCE = 1.0E-4_DP, &
                                            PRINT_FREQUENCY = 0,         &
                                            STEP_NUMBER = 100000 )
      CALL ENERGY ( PRINT = .FALSE. ) ; PE1(ITRY) = ETOTAL

      ! Directly minimized coordinates.
      CALL WRITE_STRUCTURE ( "sr_he_24_me.xyz", MEUNIT, TEMPERATURE = 0.0_DP )

      NBONDS = TMPBND   ! Reintroduce the constraints.
      ATMCRD = TMPCRD   ! Reset the coordinates.

      ! Cool the system from TSTART to TSTOP.
      CALL VELOCITY_ASSIGN ( TSTART, PRINT = .FALSE. )
      CALL DYNAMICS_OPTIONS ( 0.001_DP, 100000, 0 )
      CALL VELOCITY_VERLET_DYNAMICS (                              &
                   TARGET_TEMPERATURE = TSTART * EXP(-10.0_DP ), &
                      SCALE_FREQUENCY = 10,                       &
                      SCALE_OPTION    = "EXPONENTIAL" )
      ! Remove the constraints and reminimize.
      NBONDS = 0
      CALL ENERGY ( PRINT = .FALSE. ) ; PE2(ITRY) = ETOTAL

      ! Annealed coordinates.
      CALL WRITE_STRUCTURE("sr_he_24_ae.xyz", AEUNIT, &
                           TEMPERATURE = TSTART * EXP(-10.0_DP) )
      CALL OPTIMIZE_CONJUGATE_GRADIENT ( GRADIENT_TOLERANCE = 1.0E-4_DP, &
                                            PRINT_FREQUENCY = 0,         &
                                            STEP_NUMBER = 10000 )
      CALL ENERGY ( PRINT = .FALSE. ) ; PE3(ITRY) = ETOTAL

      ! Final (minimized sturctures starting with the annealed coordinates).
      CALL WRITE_STRUCTURE ( "sr_he_24_fe.xyz", FEUNIT,  &
                             TEMPERATURE = 0.0_DP )
   END DO

   DEALLOCATE ( TMPCRD )
   CALL WRITE_RESULTS
   CALL TIME_PRINT

   END PROGRAM SR_HE_24
```

Figure A.6: Program for performing a simulated annealing on a cluster of one strontium and 24 helium atoms (contd).

```
SUBROUTINE WRITE_STRUCTURE ( FILE, UNIT, TEMPERATURE )

CHARACTER ( LEN = * ), INTENT(IN) :: FILE
INTEGER,               INTENT(IN) :: UNIT
REAL ( KIND = DP ),    INTENT(IN) :: TEMPERATURE

INTEGER :: IATOM, IOSTAT

OPEN ( UNIT, FILE = FILE, ACTION = "WRITE", &
       POSITION="APPEND", IOSTAT = IOSTAT )
IF ( IOSTAT /= 0 ) CALL ERROR ( "WRITE_STRUCTURE", "I/O Error.", IOSTAT )

WRITE( UNIT, "(I4)" ) NATOMS
WRITE( UNIT, "(2F20.3,I6)" ) ETOTAL, TEMPERATURE, ITRY
DO IATOM = 1, NATOMS
   WRITE( UNIT, "(A,3F14.6)" ) ATMNAM(IATOM), ATMCRD(1:3,IATOM)
END DO

END SUBROUTINE WRITE_STRUCTURE
```

Figure A.7: The subroutine WRITE_STRUCTURE.

```
!   OPLS MM Definition File for Lennard-Jones Atoms
MM_Definitions OPLS_AA 1.0

! . Atom Type Definitions.
Types
! Atom Name      Atomic Number        Sigma       Epsilon
HE                       2           2.6415      0.02199
BE                       4           4.0358      0.01707
MG                      12           4.5525      0.01341
CA                      20           5.3632      0.00949
SR                      38           5.6483      0.00846
BA                      56           6.0937      0.00709
End

! . Electrostatics and Lennard-Jones Options.
Electrostatics Scale 0.5
Lennard_Jones  Scale 0.5

Units kcal/mole

! . Residue Definitions.
Residues
Residue HE
  1  0  0    ! # Atoms, Bonds and Impropers.
HE       HE       0.0

Residue BE
  1  0  0    ! # Atoms, Bonds and Impropers.
BE       BE       0.0

Residue MG
  1  0  0    ! # Atoms, Bonds and Impropers.
MG       MG       0.0

Residue CA
  1  0  0    ! # Atoms, Bonds and Impropers.
CA       CA       0.0

Residue SR
  1  0  0    ! # Atoms, Bonds and Impropers.
SR       SR       0.0

Residue BA
  1  0  0    ! # Atoms, Bonds and Impropers.
BA       BA       0.0

End
End
```

Figure A.8: The `M_He.opls` file.

```
SEQUENCE
2
SUBSYSTEM SR
1
SR 1
END
SUBSYSTEM HE_24
24
HE 24
END

END
```

Figure A.9: The `sr_he_24.seq` file.